BORN AGAIN TREES

Leo Breiman
Statistics Department
University of California
Berkeley, CA 94720
leo@stat.berkeley.edu

Nong Shang
School of Public Health
University of California
Berkeley, CA 94720
shang@stat.berkeley.edu

ABSTRACT

Tree predictors such as CART or C4.5 are often not as accurate as neural nets or use of multiple trees. But these latter methods lead to predictors whose structure is difficult to understand, whereas trees have a universal simplicity. Because of this, it is appealing to try and find tree representations of more complex predictors. We study tree representers of multiple tree predictors. These representers are larger, more stable and more accurate than trees grown the usual way. For this reason, we call them "born again" trees.

1. Introduction

A set of data (training data) T consisting of ({y(n),**x**(n)} n=1,, ... ,N) is given. Here the y(n) are either class labels in the set [1, ... ,J} (classification) or are numerical(regression) and the **x**(n) are multidimensional input vectors.. The training data T is used to consruct a predictor f(**x**) . On test set data, the predictor is accurate. However, suppose the predictor has a complex structure. For instance, the predictor could be a neural net with many hidden units. For many purposes we would like a simple picture of how the predictor operates on a given input vector **x** to produce the output prediction f(**x**) **.** One method for getting simple picture is to represent f(**x**) by a tree predictor. That is, find a tree whose output at any **x** is generally close to f(**x**) . Such trees we will call representer trees. The problem is : given a probability distribution P on the space of input variables **x** find the tree that best represents f(**x**).

Over the last two to three years, numbers of papers have come out which show that combining a multiple set of predictors, all constructed using the same date, can lead to dramatic decreases in test error Two methods seem most promising--bagging (Breiman[1995]) and arcing aka boosting (Freund and Schapire[1995], 1996]). See also Breiman[1996], Dieterrich and Kong[1995], Drucker and Cortes[1996], Quinlan[1996]. The research cited generate multiple copies of tree predictors and combined them--in regression by taking their average at an input vector **x.** In classification, the method used is to have the predictors vote for the most popular class at the point **x** . Recent work on neural nets has shown that combining a number of neural nets grown using the same data, also gives lower test set error rate.

At the of the day, what we are left with is an almost inscrutable prediction function combining many different predictors. But the resulting predictor can be quite accurate. Michie et.al.[1994]) compares the accuracy of over 18 different predictors on a variety of data sets. We looked only the larger data sets that came with set-aside test sets. Growing and combining 50 CART trees using the arcing algorithm arc-fs gives a predictor that performs much better on these four data sets than any other classifier.

But this puts us in an difficult dilemma. A single tree, whether generated by CART or C4.5 (Quinlan) has a simple and understandable structure, but combinations of many trees do not. Suppose we try to fit the data with a single layer feed forward with 25 hidden units. The

structure of the resulting neural net is also difficult to decipher. But neural nets properly fitted and regularized, can be among the most accurate of predictors.

To satisfy the needs of understanding, a single tree is needed. But the usual tree fitting methods used in CART and C4.5 produce predictors with accuracy often substantially below that of neural nets or bagged/arced trees. What is shown in this present work is that it is possible to find single trees representing predictors that have better accuracy than trees built the usual way.

The key to this construction is manufacturing data. We assume that, using the training set T, a predictor f(**x**) has been constructed such that given any input **x** of the right dimensionality, and numerical characteristics, f(**x**) outputs a predicted value of y. Then, we can manufacture a large number of inputs {**xm**}, put each one of these into f, get the predicted value ym=f(**xm**), and use the {ym,**xm**} to build a representer tree. Since there is no limit to how many **xm** we manufacture, the data used to build the representer tree can have a much larger sample size that the original training set.

There are two problems that need to be solved for this approach to work. The first is how to manufacture the **xm**. It is known that trees are consistent in the sense that, under weak restrictions, if the data consists of (y,**x**) pairs independently drawn from the same underlying distribution, then as the sample size and the size of the tree goes to infinity, its prediction error converges to the minimum possible error rate. Since we what the constructed tree to be accurate for future **x**-inputs drawn from the same distribution as the training set {**x**(n)}, we need to manufacture the **xm** from a distribution derived from that of the training set {**x**(n)}.

The second problem is that the approach, as outlined above, does not work very well. We had, early on, after putting together the arced CART classifiers, tried this approach using **xm** that were manufactured using the method detailed in Section 2. We generated data sets 10-20 times the size of the original training set and grew CART trees using this manufactured data. They were more accurate, but were so large that they could not be called simple.

The paper by Craven and Shavlik[1996] provided the piece that was essential to make representation work. Instead of generating a large manufactured data set to begin with and then grow a tree, they set a sample size NS and proceeded as follows: given an unsplit node **t** of the current representation tree, manufacture data **xm** and send it down the tree until there are NS instances in the node **t**. Now use these NS instances to split **t**. This idea solved what is the most vexing problem in tree construction--the thinning out of training data in the lower branches of the tree.

Craven and Shavlik manufacture the **xm** by using a kernel estimate for the density of each coordinate of **x**, and sample from the product of the densities. We use a different approach that preserves some of the correlation between coordinates and does not assume that the variables in x are all numerical. We also use a different "stop splitting" rule. The splitting process is stopped when the one of the two children node contains no instance of the original training set. Generally, the number of nodes generated this way is large. In order to keep tight bounds on the size of the representing trees, we assign a cost to each node and use pruning.

Section 2 describes the algorithms we use in representation. In Section 3 we give the results of testing the trees representing arced classifiers on a number of data sets. In arcing we used the Freund-Schapire algorithm (see Freund and Schapire[1995],[1996]). The tree representers are generally more accurate than the trees grown the usual way. Section 3 shows similar results for the regression case where we represent bagged trees. Section 4 gives results concerning the sizes of the representer trees, and sections 5 and 6 look at some asssociated questions.

The results in this paper seem circular. First, tree construction methods were discovered. Then, recently, it was discovered that accuracy could be improved by bagging or arcing multiple trees. Now we see that by manufacturing data and representing the multiple trees, we can get a single tree that is larger, more robust, and considerably more accurate than the original tree. Because of this, we refer to these representation trees as *born again* trees.

2. **Structure of the Algorithm.**

There are three primary components to our representations. They are a) how the **xm** are generated b) how the tree is constructed c) how the tree is pruned and a subtree selected

2.1 Generating the **xm**

The **x** in the training set and future **x** are assumed to be drawn from some underlying unknown multivariate distribution that we would like to mimic. Let **x** = (x(1), x(2), ... ,x(m)). One approach to generating the **xm** is to estimate the underlying distribution of the x using the values of the {**x**(n)} in the training set, and then sample from this estimated distribution. We take, instead, a different approach that preserves some of the correlation between variables in the input vector and allows the variables to have any characteristic-numerical or categorical.

The **xm** are manufactured as follows: a threshold number palt in [0,1] is selected (in all runs we used palt = .5 or .25). One of the {**x**(n)} in the training set is selected at random. Its coordinates are x(1,n) ,x(2,n), ... ,x(m,n). For i=1, ... ,m, a random number r in [0,1] is selected. If r > palt then xm(i)=x(i,n). Else, xm(i) is selected at random from the N values of {x(i,j), j=1, ... ,N}. We refer to this method as *100*palt% smearing.* We found, in the data sets experimented on, that 50% smearing worked well for all except one in which 25% smearing gave better results.

2.2 Constructing the born again tree

A sample size NS is set (we took NS~N, but its exact size is not critical--see remarks in Section 4). The nodes are numbered consecutively as they are formed. At the next unsplit node t, **xm** are manufactured and passed down the tree until NS land in node. These NS **xm** are passed through the arced or bagged predictor (classification or regression), to give estimated predicted values ym.

In the classification case, we get an estimate of the probability of node **t**, pr(**t**), by dividing NS by the number of times we manufactured an **xm** until we got NS of them into **t**. If p(j) is the proportion of class j among the NS cases in **t**, then set cost(**t**)= (1-max(p(j)) pr(**t**). Find the best split of **t** using the {(ym,**xm**), n=1, ... ,NS}. After the split, if either child node contains no instance of the training set, declare the parent a terminal node and leave it unsplit. In the regression case, we compute the sample variance var(**t**) of the ym in **t**, and define the cost(**t**) as pr(**t**) var(**t**).

2.3 Pruning and subtree selection.

The next step is pruning. This is done in the usual CART manner using the costs computed as mentioned above. This results in a sequence of subtrees. In the results given below, the original training set was run down the sequence of subtrees and and that subtree giving minimum error on the training set selected. This procedure sometimes selected subtrees that were larger than optimal. Another method of selecting a pruned subtree was to generated a data set from the training set using the same amount of smearing used in constructing the tree, run this set of data down the sequence of subtrees, and select the minimum error subtree. This gave smaller trees but slightly larger test set error. We give the comparisons in section 4.

3 **Experimental results--Test set error**

3.1 Classification

In testing we used a number of data sets from the UCI repository . Most of these were used in Breiman([1996a],[1996b]). Data sets in which arcing did not reduce the test set error were not used.

Table 1 Data Set Summary-Classification

Data Set	#Training	#Test	#Variables	#Classes
breast cancer	699	70	9	2
ionosphere	351	35	34	2
glass	214	21	9	6
soybean	683	68	35	19
sonar	208	21	60	2

The test set errors are given in Table 2. The first column give the results for the born again trees and the second for CART trees grown using the training set in the standard way. Comparison of the first and second columns shows the reduction in error rates during rebirth (column three). The last column gives the test set error for the 50 arced trees. The test set error estimates were gotten by deleting 10% of the data at random, growing the arced, born again and ordinary CART trees on the other 90% and then using the left-out 10% as a test set. This was repeated 100 times (50 times for the soybean data) and the test set errors averaged.

Table 2 Test Set Error (%)

Data Set	BA-CART	CART	% DECREASE	ARCED-CART
breast cancer	3.9	5.9	34	3.0
ionosphere	6.1	11.2	46	5.7
glass	28.2	30.4	7	21.6
soybean	8.4	8.6	2	6.3
sonar	25.1	32.1	22	16.0

3.2 Regression

The same data sets are used as in Breiman[1996a].

Table 3 Data Set Summary-Regression

Data Set	#Training	#Test	#Variables
Boston Housing	506	56	18
Ozone	330	33	9
Friedman#1	200	2000	10
Friedman #2	200	2000	4
Friedman #3	200	2000	4

We compare the results of the born again trees with the bagged trees and the CART results given in Breiman[1996].

Table 4 Mean Squared Test Set Error

Data Set	BA-CART	CART	%DECREASE	BAGGED -CART
Boston Housing	15.6	20.0	22	10.8
Ozone	19.0	23.9	21	18.1
Friedman#1	8.7	11.4	24	6.4
Friedman #2*	24.7	33.1	25	21.2
Friedman #3**	32.7	40.3	19	25.5

* divided by 1000 **multipliued by 1000

With the first two data sets, the same procedure was followed as in classification. That is, 10% of the data was randomly deleted, everything was run on the rest, and the deleted 10% used as a test set. This was repeated 100 times and the results averaged. The last three data sets are synthetic and in each of the 100 iterations, a 200 case training set and 2000 member test set were freshly generated. The results for bagged CART differ from the results in Breiman[1996a} mainly because 50 trees were bagged instead of 25.

4 **Tree Sizes and Parameter Settings.**

4.1 Tree Sizes

If the born again trees were too large in size, their advantage in simplicity is lost. As it turns out, the born again trees are of reasonable size--larger than CART trees, but not gigantic. We take as default the subtrees selected by running the training set down the tree. The table below gives the average number of terminal nodes in the born again trees (column 2) as compared with the average for CART trees (column 1). Another subtree selection method consists of generated a smeared data set and running it down the tree. This generally gives smaller trees (column 3) but a small increase in classification error . We give the increase as a percent of the default method missclassification rate(column 4).

Table 5 Average Number of Terminal Nodes-Classification

Data Set	CART	BA-CART	BA-CART1	ERR-INCREASE(%)
breast cancer	16	28	25	2.6
ionosphere	11	28	22	9.0
glass	18	55	45	0.9
soybean	62	74	71	1.0
sonar	10	36	33	-1.1

In the regression case, we omit the results of the alternative tree selection method since it selected trees almost identical in size to our default method.

Table 6 Average Number of Terminal Nodes-Regression

Data Set	CART	BA-CART
Boston Housing	22	58
Ozone	9	46
Friedman#1	17	49
Friedman #2	18	25
Friedman #3	22	37

4.2 Parameter Settings

We set the number NS of manufactured **xm** equal to the size of the training set in our runs. But the evidence we have is that the results, within wide limits, are not sensitive to its value. We ran the breast cancer data with NS=250 instead of the 700 used in our regular run. The results were very close. Other than NS, the only other parameter which needs to be specified is the amount of smearing.. We used 50% smearing in all data sets except in the soy bean data where 25% gave better results.

Sometimes smaller trees may be desired at the price of decreased accuracy. There are two ways to do this in our algorithm. One is to declare **t** terminal if cost(**t**) is less than some threshold value --k/N is suitable, where N is the training set size and k is small, say 1-5. Another is to declare **t** terminal if the number of training set instances in **t** is less than some number k, say k=1-5.

5. **Other issues**

A number of interesting questions come up in the context of born again trees. Some that we discuss below are fidelity of representation, computing efficiency, and accuracy.

5.1 Fidelity of representation

How faithful to the original is the representation? Put another way, how much do the born again trees differ from the arced or bagged trees they are representing. In classification, we kept track of the percent of the test set assigned different classifications by the arced trees and the born again trees. This was also averaged over the 100 iterations. Table 4 gives again (1st and 3rd columns) the ba-tree test rate and the arced-trees test rate. The 2nd column is the average percent disagreement.

Table 4 Test Set Error (%) and Difference(%)

Data Set	BA-CART	DIFFERENCE	ARCED -CART
breast cancer	3.9	2.3	3.0
ionosphere	6.1	3.3	5.7
glass	28.2	17.0	21.6
soybean	8.4	6.4	6.3
sonar	25.1	19.9	16.0

In regression, we kept track of the mean-squared differences between the bagged tree test set predictions and the born again tree predictions. Table 5 summarizes the averages of these over the iterations.

Table 5 Mean Squared Test Set Error and Difference

Data Set	BA-CART	DIFFERENCE	BAGGED -CART
Boston Housing	15.6	3.5	10.8
Ozone	19.0	2.2	18.1
Friedman#1	8.6	1.1	6.4
Friedman #2*	24.7	4.2	21.2
Friedman #3**	32.7	4.5	25.5

* divided by 1000 ** multiplied by 1000

In classification, the differences between the arced trees and their tree representers are surprisingly large--almost as large as the error rate of the arced trees. But the error rate of the born again trees is smaller than the sum of the error rate of the arced trees and the difference. The implication is that where there is a difference, there is a good chance that the born again tree has made the right choice and the arced tree the wrong one. It is odd that the imitation classifier can often be right when the classifier being imitated is wrong. The situation in regression is different. The differences between the bagged trees and their representers is small compared to the overall test set error.

5.2 Computational efficiency

It takes much longer to build a born again tree than a CART tree. The major part of the effort is not in the node splitting, but in the manufacturing of the data. If a node **t** has small probability--say, for instance, that pr(**t**)=.01, then 100 **xm** have to be manufactured for every **xm** that falls into node **t**. Thus, if NS=1000, 100,000 **xm** will need to be manufactured to get NS inputs into **t**. If the dimensionality is, say, 20, and palt=.5 then over 3,000,000 random numbers will have to generated in the manufacture of data for the single node **t.** In addition, these 100,000 **xm** will need to be passed down the tree constructed to date to see which will reach **t.**

Still, by neural net standards, the computing time is modest. For example, growing a born again tree on the soybean data takes about 50 cpu minutes on a SUN Ultrasparc 1. We recently tested a version of the algorithm which reduces the number of **xm** that need to be manufactured to get NS vectors into **t** , and cuts the computing time by a factor of three to five.

5.3 Other ways of manufacturing data

Manufacturing data by smearing seems odd at first impression. Craven and Shavlik [1996] take a more direct approach, construct a kernel density estimate for each each input variable separately, and sample from the product of the density estimates. We tried two approaches which also seemed more standard. An **x** input vector is chosen at random from the N input vectors in the training set, **x** = (x(1),x(2), ... ,x(m)). Set a value nn and for each k=1, ... ,m select the kth component of the manufactured **xm** at random from among the nn nearest neighbors of x(k) in the set of training values x(k,n), n=1, ... ,N. This is repeated as many times as necessary to get NS **xm** into the node being split. A variant of this is to select the kth component of **xm** from among the nn nearest neighbors, but with the probability of selection decreasing as the neighbor is further away. To our surprise, neither appoach did as well as smearing.

5.3 Comments on accuracy

In the classification data sets we ran, born again trees have error rates averaging 22% less than CART trees--in regression, 42% less. The idea that we can significantly improve the accuracy of trees by this roundabout method is interesting. In previous research (Shang and Breiman [1996]) we saw that tree accuracy could be improved by using the training set to estimate the (Y,**X**) distribution and then using the estimated distribution to grow the tree. While this approach also gives improved accuracy, it requires a complex estimation procedure and grows larger trees.

Breiman[1996a,b] noted that an important source of error in methods like trees was their instability, which led to high variance. Combining many trees by devices like bagging or arcing can lead to substantial reduction in variance, while leaving bias about the same. The reason that born again trees are more accurate is that they are representers of the more stable bagged or arced tree predictors. If the training set changes slightly, the CART tree may change

substantially, but not the combined tree predictor. The increased stability of the combined tree predictor is passed on to its representer.

But caveats exist. The degree of improvement is data set dependent. It is relatively small in the soy bean and glass data sets. We conjecture that the source of the problem is that these are multiple class data sets with 19 classes in the soy bean data and 6 in the glass. Then, to improve accuracy, the born again trees must give more accurate representations of 19 or 6 class distribution surfaces. Note that the born again regression trees are fairly faithful representers of the bagged trees --but here they only need to give a better representation of one prediction surface. The other classification data sets have only two classes, and so may be easier to approximate.

Another caveat is that born again trees are not automatically more accurate than CART trees. For instance, growing the born again tree on the soybean data using palt=.5 resulted in a born again tree with test set error rate higher than the CART tree. Dropping palt to .25 gave better results. Another fact we observed is that if we grow larger trees the error rate can be reduced. For instance, with the sonar data we grew trees by setting a low threshold th and splitting as long as cost(**t**) < th. In 100 iterations the average test set error was 23.3 compared with the 25.1 rate listed above. But the average number of terminal nodes went from 36 to 71.

5.4 How accurate can trees be?

There are consistency results for trees that show, under some restrictions, that as the size of the training set grows large the tree test set error converges to the Bayes rate (Breiman, et.al[1984]). Theory is all well and good, but we wanted to see what the practical limits on accuracy were using a method similar to that of the born again trees. To do this, we used the synthetic 3 class, 21 dimensional waveform data (Breiman, et.al.[1984]). For waveform training sets of size 300, the CART error rate is 29.0%, arcing 50 trees has error rate 17.8%, and the Bayes rate is 13.2%.

We constructed a large tree as follows: to split a node **t** of the tree we generated new waveform data and fed them down the tree until there were 500 instances in **t.** These 500 were used to split **t**. We declared a node terminal if its probability was less than 1/4000. The tree was pruned using the estimated gini costs in each node. A 3000 member test set was generated and poured down the pruned subtrees.

Figure 1 is a graph of the test set missclassification rate versus the number of terminal nodes. The graph indicates an asymptotic error rate of 22% for the trees constructed as described above. This indicates that an error rate substantially above the Bayes error rate is a lower limit to the best we can do using splits constructed from a fixed number of instances in each node.

6 Final remarks

More work remains to be done in the area of representing predictors by trees. We have described one method, but believe that it can be substantially improved. In particular, the method of manufacturing data needs more examination. We are perplexed by the lack of fidelity in classification and hope that others will examine this problem and find ways to improve fidelity.

L. Breiman recently gave a talk at a 1996 IMS-AMS-SIAM Summer workshop titled "Heisenberg's Principle in Statistics". The thesis was that there was a constant tradeoff between accuracy and simplicity. The most accurate predictors, for seriously high-dimensional data sets, had inscrutable structures--it was diifficult to know what pushed the prediction. But simpler predictors, like trees, usually had less accuracy.

Representing a complex prediction rule by one with more understandable structure fulfills a common need. An acquaintance working for a large investment house predicting market performance using neural nets relates that the money managers constantly ask "but what factors are driving the prediction". To answer, they fit trees to the neural net outputs. This question is not a new one to the neural net community and there are other works in the literature which attempt to derive sets of rules to represent nets. References are in the Craven and Shavlik article.

These representations do not resolve the Heisenberg dilemma--the born again trees, while they are more accurate than the CART trees, are not as accurate as the predictors they represent. For instance, the born again classification trees have, on average, 22% lower error rate than then CART trees. But the arced trees have error rates 23% lower than the born again trees. The corresponding numbers in regression are 22% and 20%. While the error rates of the born again trees can be lowered by using larger trees, this results in increasingly complex predictors and defeats the purpose of representation.

Some of the gap between the representation error rate and error rate of the represented predictor may be removed by improved representation methods. But experiments such as the one using the waveform data reported on in Section 5.4 convince us that for some data distributions, reasonable sized born again trees will have an error rate substantially above that of the predictor being represented. One can see this most clearly in regression. Suppose the response y equals a linear function f(**x**) plus noise, and suppose further that the predictor we want to represent by a tree is exactly f(**x**). If **x** is high dimensional, then many terminal nodes will be needed for a tree to represent f(**x**) accurately.

References

Breiman, L. [1996a] Bagging Predictors , Machine Learning 26, No. 2, 123-140

Breiman, L. [1996b] Bias, Variance, and Arcing Classifiers, submitted to Annals of Statistics, ftp ftp.stat.berkeley.edu pub/breiman/arcall.ps

Breiman, L., Friedman, J., Olshen R., and Stone, C. [1984] Classification and Regression Trees, Wadsworth

Dietterich, T.and Kong, E. [1995] Error-Correcting Output Coding Corrects Bias and Variance, Proceedings of the 12th International Conference on Machine Learning pp. 313-321 Morgan Kaufmann. ftp://ftp.cs.orst.edu/~tgd/papers/ml95-why.ps.gz

Drucker, H. and Cortes, C. [1996] Boosting decision trees, Neural Information Processing 8, Morgan-Kaufmann, 479-485

Freund, Y. and Schapire, R. [1995] A decision-theoretic generalization of on-line learning and an application to boosting. http://www.research.att.com/orgs/ssr/people/yoav or http://www.research.att.com/orgs/ssr/people/schapire

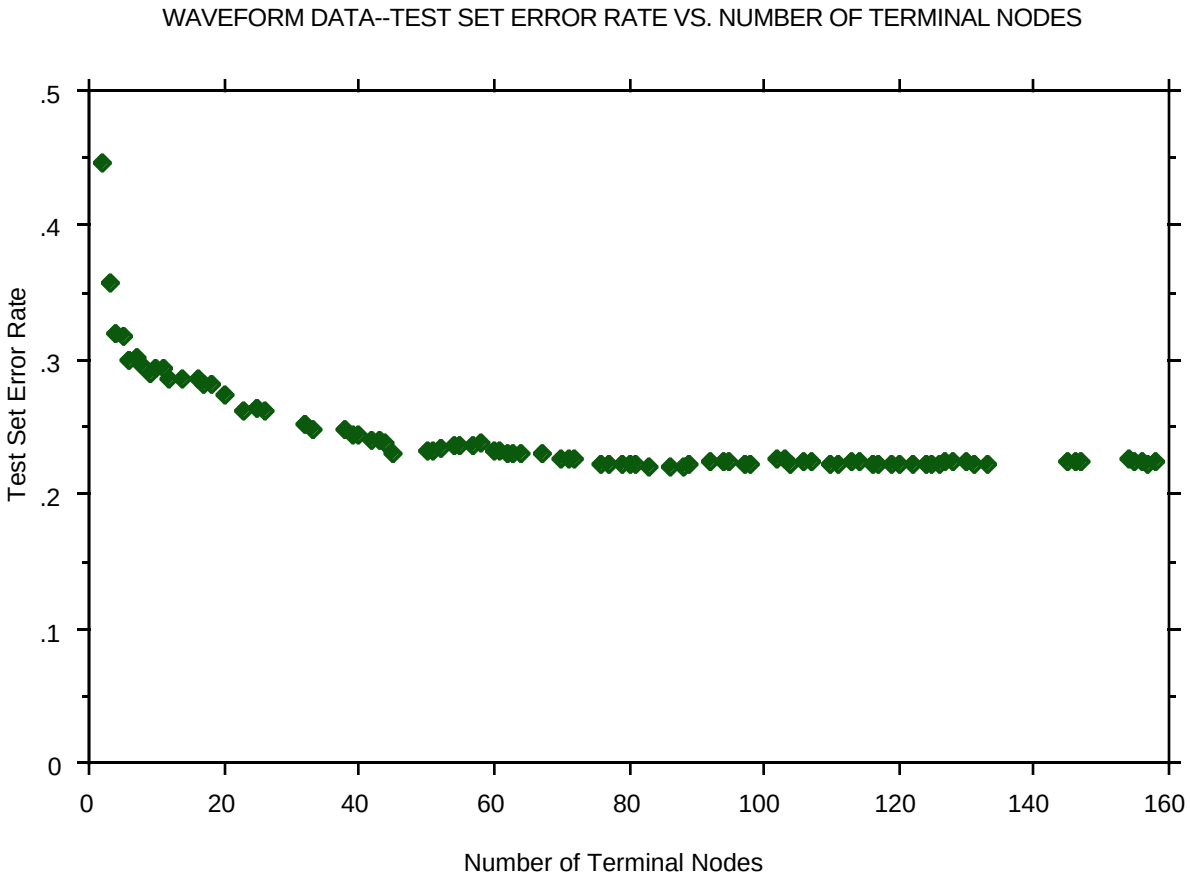
Freund, Y. and Schapire, R. [1996] Experiments with a new boosting algorithm, to appear "Machine Learning: Proceedings of the Thirteenth International Conference," July, 1996.

Craven, M and Shavlik, W. [1996] Extracting tree-structured representations of trained networks, Advances in Neural Information Processing Systems 8, 24-30

Quinlan, J. [1996] Bagging, Boosting, and C4.5, to appear in the Proceedings of AAAI'96 National Conference, on Artificial Intelligence, http://www.cs.su.oz.au/~quinlan

Shang, N and Breiman, L . [1996] Distribution based trees are more accurate, to appear, Proceedings ICONIP , September, 1996.

FIGURE 1


WAVEFORM DATA--TEST SET ERROR RATE VS. NUMBER OF TERMINAL NODES
Test Set Error Rate
.5
.4
.3
.2
.1
0
0
20
40
60
80
100
120
140
160
Number of Terminal Nodes