USING ADAPTIVE BAGGING TO DEBIAS REGRESSIONS

Leo Breiman
Statistics Department
University of California at Berkeley
Berkeley, CA 94720

Technical Report 547, February 1999

Abstract

Breiman[1996] showed that bagging could effectively reduce the variance of regression predictors, while leaving the bias unchanged. A new form of bagging we call adaptive bagging is effective in reducing both bias and variance. The procedure works in stages--the first stage is bagging. Based on the outcomes of the first stage, the output values are altered and a second stage of bagging is carried out using the altered output values. This is repeated until a specified noise level is reached. We give the background theory, and test the method using both trees and nearest neighbor regression methods. Application to two class classification data gives some interesting results.

1. **Introduction**

In regression, the average mean-squared generalization error is the sum of the noise variance, the predictor bias and predictor variance. Breiman[1996] introduced bagging as a way of reducing predictor variance. Applied to a number of real and synthetic data sets, it significantly reduced test set error. Bagging reduced variance, but had no effect on the bias, and the bias and noise variance remained as the lower limit to the prediction error. But surprisingly, there is a variant of bagging we call adaptive bagging that can reduce the bias as well as the variance. The condition for it to work is analogous to the "weak learning" condition.

Our plan of presentation is this: Section 2 gives the theoretical underpinnings for our procedure. It basically consists of running stages of baggings, but such that after each stage, the output values are modified. It will work if the basic predictor used satisfies an analogue of the weak learning condition. Section 3 introduces our implementation of the theoretical reduction procedure which we will refer to as *adaptive bagging* or *debiasing.* and tests it out by computing bias and variance for the three synthetic data sets.

Section 4 gives empirical results both on real and synthetic data sets using unpruned trees as the basic predictor. On some of the data sets adaptive bagging gives significant decreases in error rates. Computing bias and variances before and after for the synthetic data sets gives insight into when it does or does not give reductions.

In Section 5 we switch over to nearest neighbor regression to see what the effects of adaptive bagging are using this predictor. While bagging usually results in large decreases in error, debiasing produces no further error reduction on most of the data sets. We conjecture that nearest neighbor prediction satisfies the “weak learning” condition less often than trees.

In bagging, one uses the largest trees grown because bagging effects only variance. Therefore the individual predictors should be as unbiased as possible, and generally, the smaller the tree used, the larger the bias. But since adaptive bagging can reduce both bias and variance, this opens the possibility of using smaller trees and saving on computation. Section 6 explores this empirically and shows that, indeed, using adaptive bagging, much smaller trees can give error rates only slightly larger than using unpruned trees.

One interesting avenue opened by debiasing leads to classification. A two class problem can be translated into a regression problem with a zero-one output. Can adaptive bagging be applied to get effective classification? The answer is generally yes, but with a twist. Unpruned trees, in classification as in regression, have small bias. To get optimal results in classification, we need to use smaller trees with a small number of terminal nodes. Then misclassifications rates competitive with Adaboost are produced. These results, are contained in Section 7.

Most of the main recent work in regression has focused on reducing variance. The exceptions (Drucker [1997],[1999]) consist of efforts made to adapt Adaboost to a regression context. The work by Drucker contain experimental results that show decreases in mean squared error as compared to bagging. Freund and Schapire[1996] have shown that Adaboost reduces both bias and variance in classification, so it is possible that some of the bias reduction carries over into regression. There also has been promising work on regression using Support Vector Machines, but it is not clear how this is related to debiasing.

2. **A Generous Fairy Godmother**

2.1 Recalling the Bias-Variance Decomposition

Suppose a training set

$$\mathbf{T}=\{(y_n,\mathbf{x}_n)\ n=1,\dots,N\}$$

consists of N instances independently drawn from the same underlying distribution. Let $Y,\mathbf{X}$ be random variables having the same distribution. We can always write Y as:

(1) $$Y = f^*(\mathbf{X})+\varepsilon$$

where

$$f^*(\mathbf{X}) = E(Y|\mathbf{X}), \quad E(\varepsilon|\mathbf{X})=0$$

Equation (1) decomposes Y into its structural part $f^*(\mathbf{X})$ which can be predicted in terms of $\mathbf{X}$, and the unpredictable noise component. Suppose there is a prediction algorithm which uses the training set to form predictions $f(\mathbf{x},\mathbf{T})$ of y given the input vector $\mathbf{x}$. The mean-squared generalization error of f is defined as

(2) $$PE(f(\bullet,\mathbf{T}))=E_{Y,\mathbf{X}}(Y-f(\mathbf{X},\mathbf{T}))^2$$

where the subscripts indicate expectation with respect to $Y,\mathbf{X}$ holding $\mathbf{T}$ fixed.

Take the expectation of (2) over all training sets of the same size drawn from the same distribution and denote this average generalization error by $PE^*(f)$. Also, let $\bar{f}(\mathbf{x})$ be the average over training sets of the predicted value at $\mathbf{x}$. That is;

(3) $$\bar{f}(\mathbf{x})=E_{\mathbf{T}}(f(\mathbf{x},\mathbf{T})).$$

Then the bias-variance decomposition (see Breiman [1996], Geman et.al [1992]) is

(4) $$PE^*(f)=E\varepsilon^2+E_{\mathbf{X}}(f^*(\mathbf{X})-\bar{f}(\mathbf{X}))^2+E_{\mathbf{X},T}(f(\mathbf{X},\mathbf{T})-\bar{f}(\mathbf{X}))^2$$

where the first term is the noise variance, the second is the bias and the third is the variance.

In the theoretical setting for bagging (Breiman [1996]) we noted that if there was a fairy godmother kind enough to give us an endless supply of training sets of the same size, each drawn from the same distribution, then we could compute the predictor $\bar{f}(\mathbf{x})$ which has zero variance but the same bias as $f(\mathbf{x},\mathbf{T})$. Then bagging was introduced as a way of imitating the endless supply of replicate training sets to produce a reduced variance predictor.

2.2 A Better Fairy Godmother

While bagging can significantly reduce variance and, in consequence, the generalization error, it has little effect on the bias. Therefore, we postulate a more generous fairy godmother. For any prediction algorithm , denote the function $\bar{f}$ by $\mathbf{S}f^*$, where $\mathbf{S}$ is an operator on functions and $f^*$is the structural component of the output . After the end of the first round of training sets donated by the fairy godmother, we can compute $\mathbf{S}f^*$.

As an example of what the operator $\mathbf{S}$ looks like, consider nearest neighbor regression where $f(\mathbf{x},\mathbf{T})$ is the value of y corresponding to the $\mathbf{x}_n$ in $\mathbf{T}$ closest to $\mathbf{x}$. Look at the instances in $\mathbf{T}$ as N random vectors independently drawn from the $Y,\mathbf{X}$ distribution. Let $p(d\mathbf{z}|\mathbf{x})$ be the distribution of the nearest neighbor in $\mathbf{T}$ to $\mathbf{x}$. Then,

(5) $$\mathbf{S}f^*(\mathbf{x})=\int f^*(\mathbf{z})p(d\mathbf{z}|\mathbf{x})$$

For nearest neighbor regression $\mathbf{S}$ is a linear operator. This is also true for linear regression on any set of basis elements. But for predictors such as trees and neural nets, $\mathbf{S}$ is non-linear and complex.

Subtract $\mathbf{S}f^*(\mathbf{X})$ from Y getting the revised output distribution corresponding to

$$Y' = f^*(\mathbf{X}) - \mathbf{S}f^*(\mathbf{X}) + \varepsilon$$

Now, our more generous godmother supplies us with an infinite number of training sets sampled from the new Y distribution and the same $\mathbf{X}$ distribution. This lets us compute $\mathbf{S}(f^*-\mathbf{S}f^*)$. Subtract this from the Y′ distribution getting the new output distribution

$$Y'=f^*(\mathbf{X})-\mathbf{S}f^*(\mathbf{X})-\mathbf{S}(f^*(\mathbf{X})-\mathbf{S}f^*(\mathbf{X}))+\varepsilon$$

or

$$Y'=(\mathbf{I}-\mathbf{S})^2 f^*(\mathbf{X})+\varepsilon$$

The new predictor is

$$\mathbf{S}f^* + \mathbf{S}(f^*(\mathbf{X}) - \mathbf{S}f^*(\mathbf{X}))$$

Continuing this way for K steps, the current output distribution is

(6) $$Y^{(K)} = (\mathbf{I}-\mathbf{S})^K f^*(\mathbf{X}) + \varepsilon$$

and the current predictor is:

(7) $$(\mathbf{I}-(\mathbf{I}-\mathbf{S})^K)f^*(\mathbf{X}).$$

The variance in the predictor (7) is still zero--the bias is

(8) $$E_{\mathbf{X}}((\mathbf{I}-\mathbf{S})^K f^*(\mathbf{X}))^2$$

Results (6), (7), (8) hold for $\mathbf{S}$ non-linear. There are reasons to believe that (8) may go to zero for reasonable prediction methods and initial functions. For instance, , if there exists an $\varepsilon > 0$ such that for f any one of the functions $(\mathbf{I}-\mathbf{S})^K f^*(\mathbf{X})$

(9) $$E_{\mathbf{X}}((\mathbf{I}-\mathbf{S})f(\mathbf{X}))^2 < (1-\varepsilon)E_{\mathbf{X}}(f(\mathbf{X}))^2$$

the bias goes to zero as $(1-\varepsilon)^K$. In expanding the left-hand side of (9), the condition becomes that

(10) $$2E_{\mathbf{X}}(f(\mathbf{X})\mathbf{S}f(\mathbf{X})) > \varepsilon E_{\mathbf{X}}(f(\mathbf{X}))^2 + E_{\mathbf{X}}(\mathbf{S}f(\mathbf{X}))^2$$

An intuitive version of (10) is that the correlation between f and $\mathbf{S}f$ is greater than $.5(1+\varepsilon)$. This forms an interesting analogy to the concept of a weak learner in classification.

Generally, our experiments have showed that $(\mathbf{I}-\mathbf{S})f^*(\mathbf{X})$ is small. This is the decrease brought about by bagging. The problem is with the further iterates.

2.3 Debiasing as an Stage-wise Process

To set the framework for out implementation of the debiasing idea, it's useful to frame the above procedure as a succession of stages. Let $Y^{(j)}$

be the output distribution in the jth stage, with $Y^{(1)}=Y$. Abusing notation slightly, if $Y'=f+\varepsilon$, let $\mathbf{S}Y'=\mathbf{S}f$. Then,

11) $$Y^{(j+1)}=Y^{(j)}-\mathbf{S}Y^{(j)}.$$

Also, if $f^{(j)}$ is the predictor during the jth stage, with $f^{(1)}=\mathbf{S}Y$, then

12) $$f^{(j+1)}=f^{(j)}+\mathbf{S}Y^{(j)}$$

In Section 3, this sucession of stages is mimicked with finite data sets.

3 Adaptive Bagging.

Using training sets that are bootstrap samples from the original training set imitates a sequences of training sets independently sampled from the same underlying distribution. The question is how to imitate the iterative stages given in (11), (12). This is our implementation:

(I) *The procedure will proceed in stages, with each stage consisting of a bagging a fixed number K of predictors using the same input values but with altered output values. Let the values of the output variable used in the jth stage of bagging be `denoted by* $\{y_n^{(j)}\}$, *with* $\{y_n^{(1)}\}$ *being the initial output values.*

(II) *During jth stage, denote by* $\hat{y}_{n.k}$ *the predicted values given by the kth predictor grown, At the end of the jth bagging stage, set*

(13) $$y_n^{(j+1)}=y_n^{(j)}-av_k\hat{y}_{n,k}$$

where the average over $\hat{y}_{n,k}$ *is taken only over those k such that the nth instant is not in the kth training set.*

(III) If $\mathbf{x}$ *is an input value not in the initial training set, then the predictor* $f^{(j+1)}(\mathbf{x})$ *for* $\mathbf{x}$ *after j stages is gotten as follows: let* $f_{j,k}(\mathbf{x})$ *be the value predicted for* $\mathbf{x}$ *by the kth predictor in the jth stage. Then*

(14) $$f^{(j+1)}(\mathbf{x})=f^{(j)}(\mathbf{x})+av_k f_{j,k}(\mathbf{x})$$

Here is how this makes sense--assume we have repeated the bagging 100 times. In each bootstrap training set, about 37% of the instance will be left out. Therefore, the $\hat{y}_{n,k}$ are getting averaged over roughly 37 training sets and this average subtracted from $y_n^{(j)}$ to form the new outputs. The averaged $\hat{y}_{n,k}$ are approximations to test set values of $\mathbf{S}Y^{(j)}$evaluated at training set instances and the averages of $f_{j,k}(\mathbf{x})$ are approximations to $\mathbf{S}Y^{(j)}$ evaluated for instances not in the training set.

Then this procedure is repeated in successive cycles. The signal to stop repeating is this: if the mean sum-of-squares of the new y's is greater than 1.1 times the minimum of the mean sum-of-squares of the y's in any of the previous stages, then stop and use the predictor given by the end of the stage having minimum mean sum-of-squares. with the previous predictor. The reason for this stopping rule is that the mean-sum-of-squares of the y's is a measure of the residual bias (see (5)). When they begin increasing, the debiasing is beginning to reflect noise. Then we stop and go back to the minimum value.

4 Empirical Results

4.1 Bias and Variance for Synthetic Data

For synthetic data, we can compute bias and variance for any given prediction method. We work with four synthetic data sets. . The last three last originated in Friedman [1991] and are also described in Breiman[1998]. The Peak20 data (Breiman[1994]) are gotten as follows: let r=3u where u is uniform on [0,1]. Take $\mathbf{x}$ to be uniformly distributed on the 20-dimensional sphere of radius r. Let y=25exp($-.5r^2$). Training sets of size 400 are generated from the Peak20 data, and of size 200 for the Friedman data.

Using these four synthetic data sets , bias and variances were computed using unpruned CART as the predictor. The results are given in Tables 1 together with the theoretical noise variance.

Table 1 Bias and Variance--Unpruned CART

Data Set	Bias	Variance	Noise
Peak20	10.5	33.5	0.0
Friedman#1	3.4	10.7	1.0
Friedman#2+3	1.0	26.7	16.0
Friedman#3 -3	8.0	33.8	11.1

* The +and - following a data set name indicate the power of 10 to multiply the result by. For instance, the error of Friedman #3 is 8.0x10^{-3}. The same multipliers will be used in all further tables.

Then we computed bias and variances for the same data using bagging (50 unpruned trees) and adaptive bagging (fifty unpruned trees per stage). These are shown in Table 2. The - - - symbol means that adaptive bagging produced no change from bagging i.e. debiasing stopped after one stage.

Table 2 Bias-Variance for Bagging and Adaptive Bagging

Data Set	Bagging		Adaptive	
	Bias	Variance	Bias	Variance
Peak20	10.7	2.2	1.1	2.7
Friedman#1	3.8	1.4	1.2	1.9
Friedman#2	0.7	4.6	- - -	- - -
Freidman#3	7.6	5.9	6.2	7.4

In the first two data sets, adaptive bagging raises the variance slightly, but drastically cuts the bias. In Friedman #2, the bias is so small already that debiasing has no effect. Friedman#3 has bias and variance nearly equal after bagging. The debiasing occasionally goes on for two stages, which may explain the decrease in bias. But this is offset by the increase in variance.

The noise variance of Friedman#3 is 11.1, and this may make the debiasing noisy. To explore this last possibility Friedman#3 was run without noise to give bias and variance values. After bagging, the bias and variance were 7.9 and 3.6. After adaptive bagging they were 2.4 and 4.9. So it is clear that the presence of noise of the same order as the bias and variance, in this example, prevented effective debiasing.

The greatest effect is seen in the Peak20 data. Starting with a test set mean-squared error of 44.0 , bagging reduces it to 12.9, and adaptive bagging to 3.8.

4.2 Experiments on More Data Sets

Nine data sets are used in our experiments, including the four synthetic data sets used above. A summary is given in Table 3.

Table 3 Data Set Summary

Data Set	Nr. Inputs	#Training	#Test
Boston Housing	12	506	10%
Ozone	8	330	10%
Servo -2	4	167	10%

Abalone	8	4177	25%
Robot Arm -2	12	15,000	5000
Peak20	20	400	4000
Friedman#1	10	200	2000
Friedman#2 +3	4	200	2000
Friedman#3 -3	4	200	2000

Of these data sets, the Boston Housing, Abalone and Servo are available at the UCI repository. The Robot Arm data was provided by Michael Jordan.

Bagging and adaptive bagging using unpruned CART with 50 trees per stage were applied to the data sets listed in Table 3. The first three data sets listed are moderate in size and test set error was estimated by leaving out a random 10% of the instances, running both bagging and adaptive bagging on the remaining 90% and using the left out 10% as a test set. This was repeated 100 times and the test set errors averaged. The abalone data set is larger with 4177 instances and 8 input variables. It originally came with 25% of the instances set aside as a test set. We ran this data set leaving out a randomly selected 25% of the instances to use as a test set, repeated this 10 times and averaged

The 3rd column of Table 4 gives the percent reduction in test set mean-squared error from that of bagging. The last column gives the average number of stages used in the adaptive bagging.

Table 4 Percent Reduction of Error and Stages Used

Data Set	Bagging	Debiased	%Reduction	#Stages
Boston Housing	12.7	10.8	14	2
Ozone	17.8	17.8	0	1
Servo -2	24.5	25.1	-3	1
Abalone	4.9	4.9	0	1
Robot Arm -2	4.7	2.8	41	3
Peak20	12.8	3.7	71	3
Friedman #1	6.3	4.1	35	2
Friedman #2+3	21.5	21.5	0	1
Friedman #3 -3	24.8	24.8	0	1

Table 4 shows that significant reductions in error can be made by using adaptive bagging. When it does not produce reductions compared to bagging, it stops after one stage so that accuracy does not deteriorate. The slight deterioration in the servo data set is caused by the fact that the data set is small (167 instances) and the mean sum-of-squares of the y's noisy. The result is that in a small number of the 100 runs it goes to two stages where the accuracy is less than if it had stopped at one stage.

5 Nearest Neighbor Bagging and Debiasing

The decreases in prediction error using adaptive sampling depend on the prediction algorithm. In this section, nearest neighbor prediction is used. Given a training set **T** and a point **x**, the prediction for **x** is the value corresponding to that in the training set closest to **x** . Euclidean distance was used, with coordinates normalized by their standard deviations. This prediction method has bias and variance given in Table 5.

Table 5 Bias and Variance--Nearest Neighbor

Data Set	Bias	Variance	Noise
Peak20	51.7	15.7	0.0
Friedman#1	5.2	13.2	1.0
Friedman#2	2.7	34.9	16.0
Friedman#3	15.5	39.7	11.1

Although nearest neighbor regression is not a good prediction method in high dimensions (Friedman[1997]), the results of bagging and adaptive bagging are interesting. We began by running both bagging and adaptive bagging on the synthetic data sets and computing the bias and variance given by each method. The results are summarized in Table 6.

Table 6 Bias and Variance for the Nearest Neighbor Prediction Method

Data Set	Single		Bagged		Adaptive Bagging	
	bias	variance	bias	variance	bias	variance
Peak20	51.7	15.7	62.0	6.4	9.7	17.7
Friedman #1	5.2	13.2	5.2	5.8	- - -	- - -
Friedman #2	2.7	34.9	3.1	14.3	- - -	- - -
Friedman #3	15.5	39.7	15.8	17.4	- - -	- - -

Although bagging lowers the variance considerably in all four data sets, adaptive bagging does not help in the last three. In Peak20, where the bias dominates, there is a major reduction in bias more than offsetting the increase in variance. The inability of adaptive bagging to reduce bias, say in Friedman #3, may be due to violation of the “weak learner” condition.

In the next experiment, the real data sets listed in Table 1 were used. Table 7 below gives the test set error estimated by leaving out 10% of the data for use as a test set and averaging over 100 repetitions.

Table 7 Mean-Squared Error for Nearest Neighbor Prediction

Data Set	Single	Bagged	Adaptive Bagged
Boston Housing	18.6	10.5	- - -
Ozone	32.2	22.4	- - -
Servo	83.5	43.9	44.4
Abalone	6.4	6.4	- - -
Robot Arm	22.3	13.2	- - -
Peak20	67.4	68.3	27.4
Friedman#1	19.4	12.0	- - -
Friedman#2	53.6	33.4	- - -
Friedman#3	66.3	44,3	- - -

The Peak20 data consists of 400 instances in 20 dimensions and the bias overwhelms the variance. So it is not entirely surprising that bias reduction has an important effect.

For all the other data sets bias reduction is virtually non-existent, but variance reduction, via bagging, decreases error significantly. The easiest explanation is that nearest neighbor regression is not a "weak learner" for the data sets used. But whether this is the full explanation is uncertain. On all data sets the debiased CART errors were lower than the nearest neighbor debiased errors except for the Boston Housing data.

6. **Using Small trees as Predictors.**

An interesting possibility occurs with the use of debiasing. Simple predictors that would never be used because of their high bias may become fairly accurate when debiased. The trade off is between the bias and variance of the predictors used. Unpruned CART is the lowest bias and highest variance version of CART. Unpruned CART is the right thing to use with bagging, since bagging reduces variance only.

Suppose we used smaller trees. This would reduce variance and increase bias. But since the debiasing algorithms are a combination of bagging and bias removal, there might be an interesting trade-off between smaller tree size and increased debiasing. We study this question in this section. In particular, we show that with debiasing quite small trees lead to predictors only slightly less accurate than using the unpruned tree.

The advantage of using smaller trees is not that there is a significant reduction in generalization error, but that computations are can be speeded up. For instance, computing the one split leading to a stump (two terminal tree) takes about $1/\log_2 N$ as much computing time as growing the full tree, where N is the number of instances in the data set. This point was also made in the classification context by Friedman et.al [1998].

What our empirical investigation showed is that for each data set there is a smallest tree such that its error is only a bit larger than that of the largest tree. For trees smaller than this tree, the error escalates. This critical tree is usually small. Sometimes it is the stump, sometimes it is a three or four terminal node tree, and sometimes larger. I believe that the critical factor is how well the structural function can be approximated by combinations of predictors in the class.

The critical tree was found by the looking at the error as computed for Table 4 and increasing the tree size from the stump on up until the error was close enough to that of the full tree.

Table 8 shows, for each data set, the test set error of the debiased unpruned tree. This is compared to the test set error of the debiased critical tree, the average number of terminal nodes in it, and the average number of stages in the debiasing process. The errror resulting from bagging the critical tree without debiasing is given in the last column.

Table 8 Mean Squared Error for the Unpruned and Critical Trees

Data Set	Big Tree	Debiased Critical Tree			Bagged Critical Tree
	error	error	# nodes	#stages	error
Boston Housing	10.8	11.5	10	2	14.0
Ozone	17.8	19.0	4	2	21.5
Servo	25.1	30.0	5	3	57.5
Abalone	4.9	5.2	3	7	6.8
Robot Arm	2.8	3.0	100	7	6.1
Peak20	3.7	4.1	5	6	26.3
Friedman #1	4.1	4.6	2	8	16.9
Friedman #2	21.5	22.7	7	2	28.7
Friedman #3	24.8	26.7	7	2	40.6

The number of terminal nodes in the critical tree is only a fraction of the number of terminal nodes in the unpruned tree. Generally, the number of terminal nodes in the unpruned tree is a little less than the number of instances. So, for instance, in the robot arm data, the unpruned tree has about 100 times as many terminal nodes as the critical tree. The last column shows that bagging small trees without debiasing is not as effective in lowering the error rate as it is in bagging the unpruned tree.

Since there is no computational advantage in growing the largest size tree and pruning back, the smaller trees used here were constructed by fixing the number of nodes to be split. That is, my tree algorithm splits 1 into 2 and 3, 2 into 4 and 5, 3 into 6 and 7, etc. To work with trees having 4 terminal nodes, just stop after node#3 is split. This results in 4 terminal node trees that may be far from optimal among all 4 node trees. But they serve to illustrate the point. For a better method of growing small trees, see Friedman et.a.[1998].

While the error rates of the small trees come close to those of the unpruned trees, they are never less. Some accuracy is lost by going to the smaller trees. This contrasts to the case in classification, detailed in the next section.

7. Debiasing Applied to Classification

Two-class classification problems can be cast into the regression context by letting the response for class#1 be 0 and 1 for class#2. Then new data is classified as class#2 if the predicted outcome is >.5, otherwise as class#1. In our exploration of these classification problems, it became apparent that optimal accuracy was gotten by using small trees along with debiasing. This contrasts with the situation in straight regression where the optimum accuracy is always achieved by debiasing the largest tree. To illustrate, we did runs on the two class data sets listed in Table 9. These data are available in the UCI repository.

Table 9 Data Set Characteristics

#	Data Set	Instances	Inputs
1	diabetes	768	8
2	breast	699	9
3	ionosphere	351	34
4	sonar	208	60
5	heart (Clevld)	303	13
6	german credit	1000	24
7	votes	435	16
8	liver	345	6

The procedure used was this: the number of terminal nodes was varied from 2 up to 10. Keeping the number of terminal nodes fixed, 100 runs were done. In each of these runs, a random 10% of the instances were left out, the debiasing carried out, and the deleted 10% used as a test set. The test set misclassification errors were then averaged over the 100 runs. Table 10 gives the error for the two-node stump, the minimum error over the 2-10 range of terminal nodes, and, in parentheses, the number of terminal nodes at which the minimum occurred. The next column is the error for the debiased unpruned tree computed the same leave-out-10% way. The last column gives the Adaboost error rate as a standard of comparison.

Table 10 Misclassification Errors (%)

Data Set	Two-Err	Min-Err	UP-Err	Ada-Err
diabetes	24,1	23.4 (3)	24.6	26.6
breast	5.6	3.9 (5)	4.2	3.2
ionosphere	7.0	6.6 (8)	7.7	6.4
sonar	23.0	14.1 (8)	14.9	15.6
heart (Clevld)	15.6	15.6 (2)	18.8	20.7
german credit	25.3	23.6 (7)	24.8	23.5
votes	4.5	3.7 (10)	4.6	5.4
liver	29.6	25.9 (6)	30.4	28.7

Unlike the situation in regression, on the average, debiased trees with ten or fewer terminal nodes can have misclassification error less than the debiased unpruned tree. Averaging over the 8 data sets, the debiased unpruned trees have error rates 13% larger than the minimum over debiased trees with 2-10 terminal nodes. Note also that the minimum small tree error rates are comparable to the Adaboost error rates. It is possible that with a more optimal choice of the small trees such as that employed by Friedman et.al.[1998] even lower error rates could be gotten.

We also kept track of the average number of stages used for each node size. These values are given in Table 11.

Table 11 Average Number of Stages Used

	Number of Terminal Nodes					
	2	4	6	8	10	UP
Data Set						
diabetes	4.1	3.0	2.6	2.2	2.0	1.8
breast	7.1	3.0	2.4	2.1	2.0	1.7
ionosphere	4.5	3.0	2.9	2.7	2.1	2.0
sonar	3.8	3.2	3.2	3.0	3.0	2.9
heart (Cleveland)	3.8	3.0	2.4	2.0	2.0	1.2
german credit	4.2	3.1	3.0	3.0	2.8	2.0
votes	3.5	2.0	2.0	2.0	2.0	1.0
liver	3.3	3.0	2.8	2.3	2.1	1.3

Table 11 shows that the smaller and more biased the tree, the more stages are used in an effort to remove the bias. The debiasing is fairly successful even in the presence of high bias, since, for instance, Table 10 shows that the error rate in the two node stump is usually comparable to the minimum error rate. That debiasing works this well is surprizing since adaptive bagging was constructed in a regression context and the application to classification came as an afterthought.

9 Discussion

For decades, the prevalent philosophy of constructing good predictors was to grow a sequence of predictors through regularization and try to find the predictor in the sequence that had the lowest sum of bias and variance. But for a class of predictors, the given was that there was no way of reducing bias without increasing variance and vice versa. Now this thinking needs to change because both bias and variance can be simultaneously reduced.

Bagging reduces variance by altering the distribution of instances in the training set, but leaving the input and output variables in each instance unchanged. Adaptive bagging forms bootstrapped training sets but also changes the output variables in each training instance.

The key to the bias reduction is that the estimates based on the times that instances were out-of-bag can imitate independent test set estimates. This is a powerful feature of bagging. A analogous device in classification is used in Breiman [1998a] to estimate which instances in the training set will be misclassified by the aggregated classifier. The result is a classifier which is competitive with Adaboost.

Both in classification and regression, the core problem is the same. Bagging can reduce variance but is helpless against bias. Freund and Schapire[1996] have shown that Adaboost is effective in reducing bias as well as variance, but the mechanism by which it does this is still obscure. Adaptive bagging, both in regression and classification has an intuitive foundation--use the out-of-bag instances to emulate an independent test set.

The emulation is imperfect because the out-of-bag estimates govern the future course of the adaptive bagging. Thus there is some dependence between the estimates and the predictors selected. This dependence makes the out-of-bag estimates of error biased, in contrast to straight bagging where they can provide unbiased estimates of the generalization error (Wolpert and Macready [1996], Tibshirani[1996], Breiman[1997]). However, our empirical results show that for the purpose of bias reduction they work well,

References

Breiman, L. [1993] Hinging Hyperplanes for Regression, Classification and Noiseless Function Approximation, IEEE Transactions on Information Theory,39,999-1013

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

Breiman. L. [1997] Out-of-Bag Estimation, Technical Report, Statistics Department, University of California

Breiman, L. [1998] Arcing Classifiers, discussion paper, Annals of Statistics, 26, 801-824

Breiman, L.[1998a] Half and Half Bagging and Hard Boundary Points, Technical Report, Statistics Department, University of California

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

Drucker, H. [1997] Improving regressors using boosting technigues, Proceedings of the international Conference on Machine Learning, 107-115

Drucker, H. [1999] Boosting using neural networks, in Combining Artificial Neural Nets, Springer, 51-77

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

Friedman, J. [1991] Multivariate Adaptive Regression Splines, Annals of Statistics, 1991.

Friedman,J. [1997] On bias, variance, 0/1 loss, and the curse of dimensionality, J. of Data Mining and Knowlege Discovery, 1,55.

Friedman, J., Hastie, T. and Tibshirani, R. [1998] Additive Logistic Regression: a Statistical View of Boosting. Technical Report, Statistics Department, Stanford University

Geman, S., Bienenstock, E., and Doursat, R.[1992] Neural networks and the bias/variance dilemma. Neural Computation 4, 1-58

Tibshirani, R.[1996] Bias, Variance, and Prediction Error for Classification Rules, Technical Report, Statistics Department, University of Toronto

Wolpert, D.H. and Macready, W.G.[1996] An Efficient Method to Estimate Bagging's Generalization Error, in press, Machine Learning