Machine Learning, 45, 261–277, 2001

Using Iterated Bagging to Debias Regressions

LEO BREIMAN leo@stat.berkeley.edu
Statistics Department, University of California at Berkeley, Berkeley, CA 94720, USA

Editor: Robert Schapire

Abstract. Breiman (*Machine Learning, 26*(2), 123–140) showed that bagging could effectively reduce the variance of regression predictors, while leaving the bias relatively unchanged. A new form of bagging we call iterated 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 simple rule stops the process. The method is tested using both trees and nearest neighbor regression methods. Accuracy on the Boston Housing data benchmark is comparable to the best of the results gotten using highly tuned and compute- intensive Support Vector Regression Machines. Some heuristic theory is given to clarify what is going on. Application to two-class classification data gives interesting results.

Keywords: regression, bagging, out-of-bag, unbiased residuals

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 little effect on the bias, and the bias and noise remained to limit the prediction accuracy. But there is a variant of bagging we call iterated bagging that can reduce the bias as well as the variance.

Section 2 gives an intuitive overview of my procedure. It basically consists of running iterated baggings, but such that after each iteration the output values are modified. Section 3 introduces our implementation of the procedure which we will refer to as *iterated bagging* or *debiasing*.

Section 4 recalls the bias-variance definitions and computes bias and variance for bagging and iterated bagging on synthetic data sets using unpruned trees as the base predictor. Computing bias and variances before and after for the synthetic data sets gives insight into when iterated bagging does or does not give error reduction. Section 5 gives empirical error results on both on real and synthetic data sets. On some of the data sets iterated bagging gives significant decreases in error rates. Section 5.1 gives a comparison to Support Vector Regression Machines on the Boston Housing Data benchmark.

Section 6 gives some heuristic theory for iterated applications of a classifier. In Section 7 we switch over to nearest neighbor regression to see what the effects of iterated bagging

are using this predictor. While the first stage of bagging results in large decreases in error, iterated bagging produces no further error reduction on most of the data sets.

In bagging one uses the unpruned trees because bagging affects 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 iterated bagging can reduce both bias and variance, this opens the possibility of using smaller trees and saving on computation. Section 8 explores this empirically and shows that, indeed, using iterated 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 iterated 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. But to get optimal results in classification, we need to use trees with a small number of terminal nodes. Then misclassification rates comparable to Adaboost are produced. These results are contained in Section 9.

Most of the recent work in regression has focused on reducing variance. The exceptions consist of efforts made to adapt Adaboost to a regression context (Drucker, 1997, 1999) and use of Support Vector Regression Machines (see Vapnik, 1998; Drucker et al., 1997; Stitson et al., 1999; Scholkopf et al., 1999). The work using boosting contains 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. The work using Support Vector Regession Machines has given promising results on one data set (see Section 5.1) but needs testing on a variety of data.

Friedman (1999a, b) introduced regression methods he referred to as "gradient boosting" Iterated bagging has no connection to boosting; but there are some connections to Friedman's work. As requested by a referee, this is discussed in Section 10. The last section consists of final remarks and conclusions.

2. The basic idea

Assume regression type training set $T = \{(y_n, \boldsymbol{x}_n),\ n = 1, \ldots, N\}$ and an algorithm that uses T to construct a regression predictor $f_R(\boldsymbol{x}, T)$ for future y values. A seemingly attractive iterative method for increasing accuracy is:

1. Run the algorithm on T producing the regression predictor $f_R(\boldsymbol{x}, T)$.
2. Compute the residuals $y'_n = y_n - f_R(\boldsymbol{x}_n, T)$.
3. Run the algorithm on the training set $T' = \{(y'_n, \boldsymbol{x}_n)\}$, getting the new predictor $f_R(\boldsymbol{x}, T')$.
4. Compute residuals y''_n from the combined predictor $f_R(\boldsymbol{x}, T) + f_R(\boldsymbol{x}, T')$, form the predictor $f_R(\boldsymbol{x}, T'')$, etc.

If repeated many times, this iteration builds a new predictor $f_R(\boldsymbol{x}) = f_R(\boldsymbol{x}, T) + f_R(\boldsymbol{x}, T') + f_R(\boldsymbol{x}, T'') + \cdots$ which, hopefully, is more accurate than the first $f_R(\boldsymbol{x}, T)$. Applied to linear regression this idea fails. The predictor $f_R(\boldsymbol{x}, T')$ is identically zero. Linear regression on

the residuals produces zero coefficients. However, this idea, properly implemented, can work and leads to iterated bagging.

To analyze the idea more closely, let $y = f(\boldsymbol{x}) + \varepsilon$ where ε is the inherent zero-mean noise; and f is the structural part that we hope to estimate. The residuals have the form:

$$y'_n = \varepsilon_n + f(\boldsymbol{x}_n) - f_R(\boldsymbol{x}_n, T)$$

Now y'_n should be an estimate of the bias. But if $f_R(\boldsymbol{x}, T)$ is noisy due to high variance, then the residuals will reflect both the inherent noise and the variance noise. The information about the bias in the residuals will be contaminated. Therefore the first requirement to make the method work is:

Use a low variance regression algorithm

Here bagging is used. Even so, a problem still remains. Running bagging iteratively on successive residuals does not work. For almost any regression method, the training set residuals have a significant tendency to be too small since the basic idea of most regression methods is to make the residuals as small as possible. This implies that they are poor estimates of the "true" residuals. To define what I mean by the "true" residuals, suppose the single instance $(y_n, \boldsymbol{x}_n)$ is left out of the training set, and the algorithm run on this reduced training set. Then

$$\begin{aligned} y'_n &= y_n - f_R(\boldsymbol{x}_n, T - (y_n, \boldsymbol{x}_n)) \\ &= \varepsilon_n + f(\boldsymbol{x}_n) - f_R(\boldsymbol{x}_n, T - (y_n, \boldsymbol{x}_n)) \end{aligned}$$

is an unbiased estimate of the "true" residual at x_n.

Suppose this were done N times, each time leaving out a different instance. These residuals give unbiased estimate of the bias term. The second requirement needed to make the method work is

Use unbiased estimates of the residuals

If the only way to get unbiased residuals is to run the leave-one-out method N times, then the computing requirements are large (although ten-fold cross validation could be used). However, bagging can automatically compute unbiased residual estimates using out-of-bag estimation (defined in Section 3). Thus iterated bagging has the form:

a) Run bagging many times, say 50–100, getting $f_R(\boldsymbol{x}, T)$.
b) Using for y' the out-of-bag residuals estimates run bagging on the revised training set $\{(y'_n, \boldsymbol{x}_n)\}$ and form the new predictor $f_R(\boldsymbol{x}, T) + f_R(\boldsymbol{x}, T')$.
c) Repeat steps a) and b).

At some point, the intrinsic errors become comparable in size to the residual bias $f(\boldsymbol{x}) - f_R(\boldsymbol{x}, T) - f_R(\boldsymbol{x}, T') - \cdots$. The iterative process automatically senses this and stops.

3. Iterated bagging-more details

Iterated bagging is done as follows:

(I) *The procedure proceeds in stages with each stage consisting of 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* $\hat{y}_{n\cdot k}$ *the predicted values given by the kth predictor grown. At the end of the jth bagging stage, set*

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

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

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

$$f^{(j+1)}(\boldsymbol{x}) = f^{(j)}(\boldsymbol{x}) + a v_k f_{j,k}(\boldsymbol{x})$$

Here is how this makes sense: assume we have repeated the bagging 100 times. In each bootstrap training set, about 37% of the instances 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 $f^{(j)}(\boldsymbol{x})$ evaluated at training set instances.

This procedure is repeated in successive iterations. 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. The reason for this stopping rule is that the mean-sum-of-squares of the y's is a measure of the residual bias. When they begin increasing, the debiasing is beginning to reflect noise. Then we stop and go back to the minimum value.

Within limits the procedure is not sensitive to the choice of the threshold multiplier. I tried using 1.2 on a few data sets and the results were similar, but with slightly less accuracy than 1.1. Lowering it to 1.05 resulted in noticeably worse performance. All runs reported on here used 1.1.

4. Bias-variance results on synthetic data

4.1. Recalling the bias-variance decomposition

Suppose a training set $T = \{(y_n, \boldsymbol{x}_n),\ n = 1, \ldots, N\}$ consists of N instances independently drawn from the same underlying distribution. Let $Y, \boldsymbol{X}$ be random variables having the same

distribution. We can always write Y as:

$$Y = f(\boldsymbol{X}) + \varepsilon \tag{1}$$

where

$$E(\varepsilon \mid \boldsymbol{X}) = 0$$

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

$$PE(T) = E_{Y,\boldsymbol{X}}(Y - f_R(\boldsymbol{X}, T))^2 \tag{2}$$

where the subscripts indicate expectation with respect to $Y, \boldsymbol{X}$ holding 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^*. Also, let $\bar{f}(\boldsymbol{x})$ be the average over the training sets of the predicted value at $\boldsymbol{x}$. That is,

$$\bar{f}(\boldsymbol{x}) = E_T f_R(\boldsymbol{x}, T)$$

Then the bias-variance decomposition (see Breiman, 1996; Geman, Bienenstock, & Doursat, 1992) is

$$PE^* = E\varepsilon^2 + E_{\boldsymbol{X}}(f(\boldsymbol{X}) - \bar{f}(\boldsymbol{X}))^2 + E_{T,\boldsymbol{X}}(f_R(\boldsymbol{X}, T) - \bar{f}(X))^2 \tag{3}$$

where the first term is the noise variance, the second is the bias and the third is the variance. Thus, the bias and variance are the two target sources of error.

In the theoretical setting for bagging (Breiman, 1996) we noted that if we had an endless supply of independent training sets of the same size, each drawn from the same distribution, then we could compute the predictor $\bar{f}(\boldsymbol{x})$ which has zero variance but the same bias as the $f_R(\boldsymbol{X}, T)$. Then bagging was introduced as a way of imitating the endless supply of replicate training sets to produce a reduced variance predictor.

4.2. Bias and variance for synthetic data

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

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 8.0 × 10-3. The same multipliers will be used in all further tables.

Using these four synthetic data sets, bias and variances were computed using unpruned CART as the predictor. The results are given in Table 1 together with the theoretical noise variance. Then we computed bias and variances for the same data using bagging (50 unpruned trees) and iterated bagging (50 unpruned trees per stage). These are shown in Table 2. The --- symbol means that iterated bagging produced no change from bagging; i.e., one stage was optimal. In the first two data sets, iterated 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 iterated 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 iterated bagging to 3.8.

Table 2. Bias-variance for bagging and iterated bagging.

	Bagging		Iterated	
Data set	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+3	0.7	4.6	---	---
Freidman #3−3	7.6	5.9	6.2	7.4

5. Empirical results

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

The third 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 iterated bagging. Table 4 shows that significant reductions in error can be made by using iterated 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 residuals is noisy. The result is that in a small number of the 100 runs it goes to two stages resulting in less accuracy than if it stopped at one stage.

5.1. Comparison to support vector regression machines (SVRMs)

The Boston Housing data has been used as a benchmark in recent studies using Support Vector Regression Machines (see Drucker et al., 1997; Stitson et al., 1999; Scholkop et al., 1999). The usual procedure, introduced in Drucker et al. (1997), is to split off a test set of size 25 and average the test set mean-squared error over 100 iterations.

In using SVRMs (Vapnik, 1998) at least two parameters (size of the ε-tube and the regularization constant) have to be optimized using validation techniques. The best result

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	4,177	25%
Robot Arm −2	12	15,000	5,000
Peak20	20	400	4,000
Friedman #1	10	200	2,000
Friedman #2+3	4	200	2,000
Friedman #3−3	4	200	2,000

Table 4. Percent reduction of error and stages used.

Data set	Bagging	Iterated	%Reduction	#Stages
Boston Housing	11.4	9.7	15	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

to date is in Stitson et al. (1999), which also optimizes on the degree of the ANOVA kernel used. Some heavy computing leads to an error of $7.6 \pm .6$, where the $\pm$ value is twice the standard deviation of the errors over the 100 iterations divided by 10 (the square root of the number of iterations). Scholkopf et al. (1999) optimize over a kernel parameter and the regularization constant, but not over the size of the ε-tube. They present results for different values of ε. The smallest error listed is 8.7 ± 1.4.

We ran iterated bagging 100 times with test sets of size 25 and 100 trees grown in each stage. The error was 8.6 ± 1.2. In iterated bagging, no parameters need to be optimized. It is a completely turnkey procedure with modest computing requirements. Yet on the Boston Housing benchmark it gives errors comparable to highly tuned, compute-intensive SVRMs.

6. Some heuristic theory

While bagging can significantly reduce variance and, in consequence, the generalization error, it has little effect on the bias. To see how the method of iterated bagging works and how unbiased residuals and low variance prediction fit into the picture, we give some heuristic theory.

Assume that the training set outputs are of the form $f(\boldsymbol{x}_n) + \varepsilon$ where the noise ε is mean zero and independent of the inputs. The training set depends on the $\{f(\boldsymbol{x}_n)\}$, the $\{\varepsilon\}$, and the $\{\boldsymbol{x}_n\}$. Since the $\{\boldsymbol{x}_n\}$ will be held fixed, denote the training set by $\{f, \varepsilon\}$ and functions g of the $\{\boldsymbol{x}_n\}$ by $g(\cdot)$. Henceforth all functions considered will be defined only on the $\{\boldsymbol{x}_n\}$.

Given this training set, a prediction algorithm produces a function $f_R(\boldsymbol{x}, f, \varepsilon)$. Take the expectation of this function with respect to ε getting the function $S(\boldsymbol{x}, f)$. Considering S only at the values of the $\{\boldsymbol{x}_n\}$ it is a mapping from a function f defined on the $\{\boldsymbol{x}_n\}$ to another function defined on the $\{\boldsymbol{x}_n\}$.

Ordinarily, the residuals are of the form

$$y'_n = \varepsilon_n + f(\boldsymbol{x}_n) - f_R(\boldsymbol{x}_n, f, \varepsilon)$$

The overfitting difficulty stems from the correlations between the noise in the output and the same noise being used to construct the predictor. What we called unbiased residuals are

of the form

$$y'_n = \varepsilon'_n + f(\boldsymbol{x}_n) - f_R(\boldsymbol{x}_n, f, \varepsilon)$$

where the $\{\varepsilon'_n\}$ are independent of the $\{\varepsilon_n\}$, but with the same distribution. Running the algorithm on these outputs and adding to the initial predictor gives the predictor

$$f_R(\cdot, f, \varepsilon) + f_R(\cdot, f - f_R(f, \varepsilon), \varepsilon')$$

Take expectation of this with respect to ε' and ε. The first term becomes Sf. The second term, taking expectation with respect to ε' first, becomes

$$S(\cdot, f - f_R(f, \varepsilon))$$

The key to this step is the independence of ε and ε'. To take expectation with respect to ε, we argue as follows: for a low variance algorithm $f_R(\cdot, f, \varepsilon)$ does not vary much with ε. So make the approximation

$$E[S(\cdot, f - f_R(\cdot, f, \varepsilon))] \approx S(\cdot, f - Ef_R(\cdot, f, \varepsilon)) = S(I - S)f \tag{4}$$

where E denotes expectation with respect to ε. To make this approximation more understandable, do a Taylor expansion:

$$f_R(\cdot, f, \varepsilon)) = h^{(1)}(\cdot, f) + (\varepsilon, h^{(2)}(\cdot)) + \cdots$$

where both h's are functions on the $\{\boldsymbol{x}_n\}$ and $(,)$ indicates inner product of two N-vectors. Suppose the higher order terms can be neglected. Then

$$E[f_R(\cdot, f, \varepsilon))] = h^{(1)}(\cdot, f) = S(\cdot, f)$$

Now do the Taylor expansion

$$S(\cdot, f_R(\cdot, f, \varepsilon)) = S(S(\cdot, f) + (\varepsilon, b(\cdot)) + \cdots$$

Assuming higher order terms can be ignored and taking the expectation gives the approximation (4). Going to the next stage, the new unbiased residuals are

$$\varepsilon'' + f(\cdot) - f_R(\cdot, f, \varepsilon) - f_R(\cdot, f - f_R(f, \varepsilon), \varepsilon')$$

where the ε'' are independent of the ε', ε. Repeating the argument above gives the expected value of the augmented predictor as

$$Sf + S(I - S)f + S(I - S)(I - S)f$$

Continuing this way for K steps, gives the expectation of the predictor as

$$S\left(\sum_{k=0}^{K}(I-S)^k f\right)$$

If there is convergence—that is, if:

$$\sum_{k=0}^{K}(I-S)^k f \to f^{(\infty)} \tag{5}$$

then the limit function must satisfy the equation

$$(I-S)f^{(\infty)} = f^{(\infty)} - f$$

implying $Sf^{(\infty)} = f$. Thus, the limiting expectation of the iterated predictor is exactly the structural function f.

The open question remaining is the convergence in (5). For any collection of N real numbers $a_1, a_2, \ldots, a_N$, define

$$\|a\| = \left(\frac{1}{N}\sum_n a_n^2\right)^{1/2}$$

Weak learning condition

There is an $\varepsilon > 0$ *such that for every function* $g(\cdot)$ *defined on the* $\{\boldsymbol{x}_n\}$

$$\|(I-S)g\| \le (1-\varepsilon)\|g\| \tag{6}$$

If the weak learning condition is satisfied, then

$$\|(I-S)^k f\| \le (1-\varepsilon)^k \|f\|$$

and this is enough to guarantee convergence in (5).

The requirement (6) says that for a class of functions, the root mean-squared difference between the function and its estimate by the algorithm is uniformly slightly less than the root mean-squared value of the function. That is, the algorithm gives, uniformly, a slight edge in accuracy. This forms an interesting analogy to the concept of a weak learner in classification.

Clearly, what has been done is not rigorous, and it would take a lot of work to make it so. But it does indicate the importance of using unbiased residuals and low variance predictors. Plus it gives an indication that to work, the predictors may have to satisfy some weak learning condition.

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+3	2.7	34.9	16.0
Friedman #3−3	15.5	39.7	11.1

7. Nearest neighbor bagging and debiasing

The decreases in prediction error using iterative stages of bagged predictors depend on the prediction algorithm. In this section, nearest neighbor prediction is used. Given a training set T and a point $\boldsymbol{x}$, the prediction for $\boldsymbol{x}$ is the value corresponding to that in the training set closest to $\boldsymbol{x}$. Euclidean distance was used with coordinates normalized by their standard deviations. This prediction method has bias and variance given in Table 5. Although nearest neighbor regression is not a good prediction method in high dimensions (Friedman, 1997), the results of bagging and iterated bagging are interesting. We began by running both bagging and iterated bagging on the synthetic data sets and computing the bias and variance given by each method. The results are summarized in Table 6. Although bagging lowers the variance considerably in all four data sets, iterated 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 iterated 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 3 were used. Table 7 below gives the test set error estimated the same way as in Section 5. 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 error.

Table 6. Bias and variance for the nearest neighbor prediction method.

	Single		Bagged		Iterated bagging	
Data set	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+3	2.7	34.9	3.1	14.3	---	---
Friedman #3−3	15.5	39.7	15.8	17.4	---	---

Table 7. Mean-squared error for nearest neighbor prediction.

Data set	Single	Bagged	Iterated
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 −2	22.3	13.2	---
Peak20	67.4	68.3	27.4
Friedman #1	19.4	12.0	---
Friedman #2+3	53.6	33.4	---
Friedman #3−3	66.3	44.3	---

8. 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 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 (assuming my tree version, which sorts only once at the top; and that a floating point operation time is at least 3–4 times as large as a sorting operation time). 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 unpruned 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 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 error resulting from bagging the critical tree without debiasing is given in the last column. Generally, the

Table 8. Mean squared error for the unpruned and critical trees.

Data set	Big tree error	Debiased critical tree		Bagged critical tree	
		error	# nodes	# stages	error
Boston Housing	9.7	11.5	10	2	14.0
Ozone	17.8	19.0	4	2	21.5
Servo −2	25.1	30.0	5	3	57.5
Abalone	4.9	5.2	3	7	6.8
Robot Arm −2	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+3	21.5	22.7	7	2	28.7
Friedman #3−3	24.8	26.7	7	2	40.6

number of terminal nodes in the unpruned tree is about the same magnitude as the number of instances. So the number of terminal nodes in the critical tree is only a fraction of the number of terminal nodes in the unpruned tree For instance, in the robot arm data, the unpruned tree has about a hundred 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 four terminal node trees that may be far from optimal among all four node trees. But they serve to illustrate the point. For a better method of growing small trees, see Friedman et al. (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.

9. 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 obtained 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 unpruned tree. To illustrate, we did runs on the two-class data sets listed in Table 9. These data are available in the UCI repository. 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,

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	1,000	24
7	Votes	435	16
8	Liver	345	6

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. 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 eight 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 a more optimal choice of the small trees such as that employed by Friedman et al. (1998) could obtain even lower error rates.

The min-error rates may be biased down because of the minimum selection. But I believe that use of cross-validation to determine the best tree size would have given comparable test set errors. This was not done because this study of classification was not intended to present iterated bagging as a good classification algorithm, but only to look at its ability to reduce bias.

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	10.7
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

Table 11. Average number of stages used.

	Number of terminal nodes					
Data set	2	4	6	8	10	UP
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

I also kept track of the average number of stages used for each node size. These values are given in Table 11. 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 surprising since iterated bagging was constructed in a regression context and the application to classification came as an afterthought.

10. Connection with Friedman's work

In some recent research, Friedman (1999a) constructed another method for regression which has some similarities to iterated bagging. The idea of "boosting" the gradient comes from my paper, Breiman (1997a), which showed that Adaboost and other arcing algorithms are equivalent to doing a down-the-gradient descent on given target functions (but where the word boosting came from is a mystery to me).

In the mean-squared regression context Friedman's procedure does an iterative fitting of the residuals with the best fit chosen out of the class of trees of fixed size. But, as mentioned in the Section 2, this procedure does not work very well.

Overfit starts soon and the generalization error increases. To prevent this, a clever device is used. If $f(\boldsymbol{x})$ is the function in the tree-class that gives the best fit to the residuals at the k-stage, then instead of adding $f(\boldsymbol{x})$ to the current predictor, add $vf(\boldsymbol{x})$ where v is between zero and one. There is still the problem of overfitting when too many functions are added to the predictor. So the number of functions M in the predictor also has to be controlled. The best values of v and M are determined by cross-validation.

A later paper (Friedman, 1999b) added a stochastic element to the above procedure. Each training set was chosen to be about half the size of the original training set by sampling without replacement. The stochastic element improved performance and also added another similarity to iterated bagging.

The empirical results are good, but I cannot compare them to iterated bagging because the data sets used in Friedman's work are not available. In spite of some superficial similarities,

there is a fundamental difference between the two methods. For instance, Friedmans's approach gets its best results using small trees—not more than ten terminal nodes. Iterated bagging, at least in regression, gets its best results using the largest possible trees.

11. 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. Iterated bagging forms bootstrapped training sets but also changes the output variables in each training instance at each stage.

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. An 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. Iterated 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 iterated 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 & Macready, 1999; Tibshirani, 1996; Breiman, 1997). However, our empirical results show that for the purpose of bias reduction, they work well.

Acknowledgment

I am indebted to a trio of hard working and perceptive referees who made suggestions that greatly improved the readability of this paper and who also caught many of the errors I made.

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*(2), 123–140.

Breiman, L. (1997a). Arcing the edge. Technical Report, Statistics Department, University of California.

Breiman, L. (1997b). 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., & Stone, C. (1984). *Classification and Regression Trees*. Wadsworth.
Drucker, H. (1997). Improving regressors using boosting techniques. In *Proceedings of the International Conference on Machine Learning* (pp. 107–115).
Drucker, H. (1999). *Combining Artificial Neural Nets* (pp. 51–77). Berlin: Springer.
Drucker, H., Burges, C., Kaufman, K., Smola, A., & Vapnik, V. (1997). Support vector regression machines. *Advances in Neural Information Processing Systems, 9*, 155–161.
Freund, Y., & 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. (1999a). *Greedy Function Approximation: A Gradient Boosting Method*. Available at http://www-stat.stanford.edu/∼jhf/.
Friedman, J. (1999b). *Stochastic Gradient Boosting*. Available at http://www-stat.stanford.edu/∼jhf/.
Friedman, J., Hastie, T., & Tibshirani, R. (1998). Additive logistic regression: A statistical view of boosting. Technical Report, Statistics Department, Stanford University.
Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation, 4*, 1–58.
Scholkopf, B., Bartlett, P., Smola, A., & Williamson, R. (1999). Shrinking the tube: A new support vector regression algorithm. *Advances in Neural Information Processing Systems, 11*, 330–336.
Stitson, M., Gammerman, A., Vapnik, V., Vovk, V., Watkins, C., & Weston, J. (1999). Support vector regression with ANOVA decomposition kernels. *Advances in Kernel Methods—Support Vector Learning* (pp. 285–291). Cambridge, MA: MIT Press.
Tibshirani, R. (1996). Bias, variance, and prediction error for classification rules. Technical Report, Statistics Department, University of Toronto.
Vapnik, V. (1998). *Statistical Learning Theory*. New York: Wiley.
Wolpert, D. H. & Macready, W. G. (1999). An efficient method to estimate bagging's generalization error. *Machine Learning*, *35*(1), 41–55.

Received July 12, 1999
Revised February 23, 2001
Accepted February 26, 2001
Final manuscript February 26, 2001