Technical Report UT-AI-TR-05-321, Artificial Intelligence Lab, University of Texas at Austin

Combining Bias and Variance Reduction Techniques for Regression

Yuk Lai Suen[1], Prem Melville[2], and Raymond J. Mooney[3]

[1] Dept. of Electrical and Computer Engr., Univ. of Texas at Austin
suen@ece.utexas.edu,
[2] Dept. of Computer Sciences, Univ. of Texas at Austin
melville@cs.utexas.edu,
[3] mooney@cs.utexas.edu

Abstract. Gradient Boosting and bagging applied to regressors can reduce the error due to bias and variance respectively. Alternatively, Stochastic Gradient Boosting (SGB) and Iterated Bagging (IB) attempt to simultaneously reduce the contribution of both bias and variance to error. We provide an extensive empirical analysis of these methods, along with two alternate bias-variance reduction approaches — bagging Gradient Boosting (BagGB) and bagging Stochastic Gradient Boosting (BagSGB). Experimental results demonstrate that SGB does not perform as well as IB or the alternate approaches. Furthermore, results show that, while BagGB and BagSGB perform competitively for low-bias learners, in general, Iterated Bagging is the most effective of these methods.

1 Introduction

The decomposition of a learner's error into *bias* and *variance* terms provides a way of analyzing the behavior of different learning algorithms [1]. Various methods have been devised to reduce either the bias or variance of a learner. Some methods, such as Gradient Boosting [2], can reduce bias by increasing the expressive power of the base learner. While other methods, such as bagging [3], mainly reduce variance by subsampling the training data. There have been some attempts of combining techniques for bias and variance reduction, both for classification [4; 5] and for regression [6; 7]. For regression, Friedman [7] introduced Stochastic Gradient Boosting (SGB) as a method that reduces the variance of Gradient Boosting (GB) by incorporating randomization in the process. Breiman [6] presented a related method, Iterated Bagging (IB) that attempts to reduce the bias of bagging predictors. Despite their similarities, to our knowledge, there has been no direct experimental comparison of these two methods. In this paper, we present a detailed empirical analysis of SGB and IB. We show that IB significantly outperforms SGB when applied to both pruned and unpruned regression trees.

We also explored two alternate methods for combining bias and variance reduction techniques for regression — bagging Gradient Boosting (BagGB) and bagging Stochastic Gradient Boosting (BagSGB). Our experiments show that these methods also significantly outperform SGB. In comparison to IB, BagGB and BagSGB are equally effective when applied to unpruned regression trees. However, for pruned regression trees, which have a higher bias, we observe that IB is the most effective at error reduction. This paper also presents a bias-variance analysis of the different algorithms, which provides a better understanding of the relative effectiveness of these methods.

Section 2 provides a brief background on the bias-variance decomposition of error. In section 3 we describe all the algorithms discussed in this paper, and our main experimental results are presented in section 4. In section 5 we discuss the results of our bias-variance analysis; and section 6 presents our future work and conclusions.

2 Bias-Variance Decomposition of Error

The following formulation of the bias-variance (BV) decomposition is based on [8]. Let us assume our data arose from a model $y = F(x) + \epsilon$, where the random error ϵ has $E(\epsilon) = 0$ and $Var(\epsilon) = \sigma_\epsilon^2$. Then the expected prediction error of a regression model $\hat{F}(x)$ for a point $x = x_i$ using squared-error loss can be expressed as:

$$\begin{aligned}\Psi(y, \hat{F}(x_i)) &= E[(y - \hat{F}(x_i))^2 | x = x_i \\ &= \sigma_\epsilon^2 + [E(\hat{F}(x_i)) - F(x_i)]^2 + E[\hat{F}(x_i) - E(\hat{F}(x_i))]^2 \\ &= \sigma_\epsilon^2 + bias^2(\hat{F}(x_i)) + variance(\hat{F}(x_i))\end{aligned} \tag{1}$$

The first term is the *irreducible error*, which is the variance of the target function around its true mean $F(x)$. This error cannot be avoided no matter how well we model $F(x)$. The second term is the contribution of squared bias to error, which is the amount by which the average of our estimates differs from the true mean. The last term is the contribution of variance to error, which is the expected squared deviation of $\hat{F}(x_i)$ around its mean. For brevity, we will refer to the contribution of squared bias and variance to error as $bias^2$ and *variance* respectively. In general, more complex models have lower bias and higher variance; e.g., unpruned decision trees tend to have low bias and high variance, while decision stumps have a very high bias but low variance.

3 Algorithms

3.1 Gradient Boosting and Stochastic Gradient Boosting

Gradient Boosting [2] (GB) is an iterative algorithm which constructs additive models by fitting a base learner to the current *residue* at each iteration; where the residue is the gradient of the loss function being minimized with respect to the model values at each data point. In [9], Friedman introduced

Algorithm 1 Stochastic Gradient Boosting

```
Given:
$M$ – maximum number of stages
$\{x_n, y_n\}_{n=1}^{N}$ – training set of size $N$
$f = \frac{\tilde{N}}{N}, 0 < \tilde{N} \leq N$ – fraction parameter that determines the size of subsample
$\nu$ – shrinkage parameter
$\mathcal{L}$ – base learner

1. For $m = 1$ to $M$ do:
2.     Select random subset $\{x_{\tilde{n}}, y_{(\tilde{n},m)}\}_{\tilde{n}=1}^{\tilde{N}}$ from $\{x_n, y_{(n,m)}\}_{n=1}^{N}$
3.     Apply learner $\mathcal{L}$ to sample set $\{x_{\tilde{n}}, y_{(\tilde{n},m)}\}_{\tilde{n}=1}^{\tilde{N}}$ to produce predictor $\hat{F}_m$
4.     Replace residues of training set $\{x_n, y_{(n,m)}\}_{n=1}^{N}$ to form $\{x_n, y_{(n,m+1)}\}_{n=1}^{N}$,
       where $y_{(n,m+1)} = y_{(n,m)} - \nu \cdot \hat{F}_m(x_n)$

Output: $y = \sum_{m=1}^{M} \nu \cdot \hat{F}_m(x)$
```

Stochastic Gradient Boosting (SGB) which improves the accuracy of GB by reducing its error due to variance.

In SGB, at each iteration a subsample of data is drawn uniformly at random, without replacement, from the full training set. This random subsample is used to train the base learner to produce a model for the current stage. Friedman [7] states that the idea of using a random subset of the training set at each stages originates from bootstrap sampling in bagging, and has a similar variance-reducing effect on the combined model.

The SGB method is presented in Algorithm 1. GB can be viewed as a special case of this algorithm in which the entire training set is used at each iteration, i.e., $f = 1.0$. SGB proceeds in M stages, each of which learns a predictor $\hat{F}_m$ to predict the current residues of the training set and updates the residues of the training set for the next stage. The learning in the mth stage starts with randomly selecting $\tilde{N} = N \cdot f$ instances from the N initial training instances $\{x_n, y_{(n,m)}\}_{n=1}^{N}$, where $f \in (0 \ldots 1]$ is a user-defined fraction value. The learner $\mathcal{L}$ then uses this new training set $\{x_{\tilde{n}}, y_{(\tilde{n},m)}\}_{\tilde{n}=1}^{\tilde{N}}$ to learn a new predictor $\hat{F}_m$ that attempts to minimize the squared error $(y - \hat{F})^2$. SGB can be used to handle other loss functions, such as least absolute deviation, Huber loss and logistic binomial log-likelihood [2; 7]. However, this paper will only focus on least squared error.

The target value y_n is treated as the initial residue $y_{(n,1)}$ for the nth instance. The SGB residue of a training instance in the next iteration is the difference between the current residue and the negative gradient of the loss function, i.e., a partial differentiation of the loss function with respect to the predictor being learned in the current stage. The residue $y_{(n,m+1)}$ for the nth instance in the $(m+1)$th iteration is computed as:

$$y_{(n,m+1)} = y_{(n,m)} - \nu \cdot \hat{F}_m(x_n)$$

Algorithm 2 Iterated Bagging

```
Given:
$M$ – maximum number of stages
$K$ – number of bagging predictors in each stage
$\tau$ – threshold of mean sum-of-squares of residues
$\{x_n, y_n\}_{n=1}^N$ – training set of size $N$
$\mathcal{L}$ – base learner

1. Initialize minimum residue, $\epsilon_{M^*} = \infty, M^* = 0$
2. For $m = 1$ to $M$ do:
3.     Learn a set of $K$ bagging predictors $\{\beta_{(k,m)}\}_{k=1}^K$ with learner $\mathcal{L}$
       applied to bootstrap samples selected from $\{x_n, y_{(n,m)}\}_{n=1}^N$
4.     Calculate the residue $y_{(n,m+1)} = y_{n,m} - \sum_{\hat{k}=1}^{\hat{K}} \beta_{(\hat{k},m)}(x_n)/\hat{K}$,
       where $\beta_{(\hat{k},m)}$ is one of the $\hat{K}$ bagging predictors not trained on $x_n$
5.     Replace residues of the training set to form $\{x_n, y_{(n,m+1)}\}_{n=1}^N$
6.     Calculate the mean sum-of-squares of residues, $\epsilon_m = \sum_{n=1}^N y_{(n,m+1)}/N$
7.     If $\epsilon_m < \epsilon_{M^*}$ then $M^* = m, \epsilon_{M^*} = \epsilon_m$
8.     Exit the loop if $\epsilon_m > \tau \cdot \epsilon_{M^*}$

Output: $y = \sum_{m=1}^{M^*} \sum_{k=1}^K \beta_{(k,m)}(x)/K$
```

where ν is a user-defined shrinkage value to weigh and smooth the prediction at each stage.

After M stages, SGB creates a combined (additive) model, i.e., a set of trained predictors $\{\hat{F}_m\}_{m=1}^M$. The predicted output y for a test input x is given by:

$$y = \sum_{m=1}^M \nu \cdot \hat{F}_m(x)$$

which is a weighted sum of the outputs of the predictors in the combined model.

The performance of SGB is dependent on the fraction parameter f which determines the size of the training set at each iteration. Although experimental results in [7] suggests that there is a weak correlation between f and the error, there was no conclusive methodology provided for the selection of f.

3.2 Iterated Bagging

Bagging has been shown to reduce the variance of predictors, while leaving the bias largely unchanged [3]. Iterated Bagging (IB) [6], also known as Adaptive Bagging [10], is an effort to reduce the bias error of the low-variance bagging predictors. Similar to SGB, it is a stage-wise algorithm that attempts to minimize the residue in each stage. IB addresses bias and variance reduction in two ways: (1) it uses low-variance bagging predictors to compute residues and (2) it computes unbiased estimates of residues using out-of-bag calculations [10]. The outline of IB is presented in Algorithm 2.

The algorithm proceeds in M stages, with each stage learning K bagging predictors and re-computing the residues of all training instances. At the mth stage, a set of K bagging predictors $\{\beta_{(k,m)}\}_{k=1}^{K}$ are generated by applying the base learner $\mathcal{L}$ to bootstrap samples of size N selected randomly, with replacement, from the training instances $\{x_n, y_{n,m}\}_{n=1}^{N}$. The target value y_n is treated as the initial residue $y_{(n,1)}$ for the nth instance. After training K bagging predictors, IB updates the residue of each instance in the training set. The computation of residues in IB is very similar to that of SGB using least squared error. The residue $y_{(n,m+1)}$ of the nth instance at the $(m+1)$th stage is the difference between the current residue $y_{(n,m)}$ and the average out-of-bag prediction on the input x_n over the K bagging predictors learned at the mth stage:

$$y_{(n,m+1)} = y_{n,m} - \textstyle\sum_{\hat{k}=1}^{\hat{K}} \beta_{(\hat{k},m)}(x_n)/\hat{K}$$

where $\beta_{(\hat{k},m)}$ is one of the $\hat{K}$ bagging predictors that was not trained on x_n. The method for computing the residues when $\hat{K}$ is 0 is not mentioned in [6]. We deal with this case by updating the residue with the difference between the current residue and the mean of the output on x_n over all bagging predictors in this stage.

The IB algorithm keeps track of the minimum mean sum-of-squares of the residues ϵ_{M^*} in the M^*th stage. If the mean sum-of-squares of the residues ϵ_{m+1} for the $(m+1)$th stage exceeds $\tau \cdot \epsilon_{M^*}$, then the algorithm stops and returns the K bagging predictors in each of the M^* iterations, where τ is a user-defined threshold parameter. The prediction on a test instance is given by summing up the average of the predicted values across the K bagging predictors in the M^* stages:

$$y = \textstyle\sum_{m=1}^{M^*} \sum_{k=1}^{K} \beta_{(k,m)}(x)/K$$

In general, bagging works well with high-variance base regression learners. IB, on the other hand, improves over bagging and reduces both variance and bias of the base learner [6].

3.3 Bagging GB and Bagging SGB

In addition to SGB and IB, we explored two alternative approaches to bias-variance reduction — bagging Gradient Boosting (BagGB) and bagging Stochastic Gradient Boosting (BagSGB). BagGB and BagSGB use GB and SGB, respectively, as the base learners in each stage of building a bagging predictor. A total of K bootstrap sets of training instances are randomly selected to train K GB (or SGB) predictors. The output y of a test input x is predicted by averaging the sum of predictions of the K base predictors.

BagGB should reduce the variance error of predictions by stabilizing the predictions of the GB base learners. BagGB and IB are similar, as they both possess two components: (1) a bagging predictor to stabilize the predictions of the base learners by averaging the results of the predictors each trained with a different bootstrap sample and (2) a greedy stage-wise training of base predictors to minimize the residues. The difference between IB and BagGB is that IB

Datasets	Attr.	Ins.	Datasets	Attr.	Ins.	Datasets	Attr.	Ins.
auto93	26	93	autoHorse	26	205	autoMpg	8	398
autoPrice	26	205	bodyfat	15	252	breastTumor	9	286
cholesterol	14	303	cleveland	14	303	cloud	6	194
cpu	10	209	echoMonths	13	132	detroit	14	13
fruitfly	5	125	fishcatch	8	159	housing	13	506
hungarian	14	294	lowbwt	11	189	meta	22	528
pbc	20	418	pharynx	13	195	pollution	15	60
pwLinear	10	200	sensory	11	576	servo	5	167
strike	6	625						

Table 1. Summary of UCI datasets with the number of attributes and instances.

Algorithm	Description
IB	10 stages of IB with 10 stages of bagging each ($M = 10, K = 10$).
BagSGB	10 stages of bagging $\times$ 10 SGB iterations each ($M = 10$).
BagGB	10 stages of bagging $\times$ 10 GB iterations each ($M = 10$).
SGB	100 iterations ($M = 100$).
GB	100 iterations ($M = 100$).
Bagging	100 stages of bagging M5$'$ trees.
M5$'$	pruned or unpruned M5$'$ tree induction

Table 2. Experimental setup of each method.

performs greedy stage-wise training with a set of bagging predictors to stabilize the predictions of their base learners, while BagGB stabilizes the predictions of a set of base-predictors each of which performs greedy stage-wise training.

Although SGB already attempts to reduce the variance of GB through randomization, we believe that bagging SGB may further enhance its variance reduction. Because of the similarity between the algorithms, we would expect BagGB, BagSGB, and IB to demonstrate comparable performance.

4 Experimental Evaluation

4.1 Methodology

We ran all our experiments on 25 datasets, with continuous class (target) values, from the UCI repository [11]. Table 1 gives a summary of these datasets. They vary in the number of attributes and the number of instances, thus providing a diverse test bed for evaluating the performance of different algorithms.

We compared the 7 different regression methods listed in Table 2. The table provides the setup parameters for each of the methods used. The performance of most meta-learners (additive models) varies with the number of base models used. In order to make the comparison fair, we chose parameters such that each method produces 100 base models. In the case of IB, this is an upper bound, since it can choose to use fewer models. As a base learner for all the meta-learners we used M5$'$ [12], which is regression tree induction modified based on [13] and [14]. We ran separate sets of experiments on pruned M5$'$ and unpruned M5$'$. In pruned M5$'$, the regression tree is pruned back from the leaves, so long as the expected estimated error decreases. All our results were averaged over 10 runs of 10-fold cross-validation. The difference in performance between two systems was compared using a two-tailed paired t-test ($p < 0.05$).

The performance of SGB and BagSGB is dependent on the fraction parameter f chosen for the experiment. Some values for f perform significantly better than others on the same dataset. In order to compare with the best instances of SGB and BagSGB, we performed 10 runs of 10-fold cross-validation on SGB and BagSGB with different values of f from $\{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ and selected the f that produced the lowest error for each dataset. Note that often the lowest error is not statistically significantly different from the error produced by other values of f.

4.2 Results

Tables 3(a) and 3(b) summarize the results of our experiments using unpruned and pruned M5$'$ base learners respectively. Each cell in the tables report a comparison between the algorithm in the row versus the algorithm in the column in terms of win/draw/loss records. The win/draw/loss record presents three values, the number of data sets for which algorithm A obtained better, equal, or worse performance than algorithm B with respect to root-mean-squared error. A win or loss is only counted if the difference in values is determined to be significant at the 0.05 level by a paired t-test. The last column of each table presents the percentage reduction of the root-mean-square error using different algorithms compared with using M5$'$. This value is averaged over all the 25 datasets, and provides an indication of the magnitude of improvements one can expect on average.

We also ran experiments (not presented in the table) comparing the base learners — unpruned M5$'$ with pruned M5$'$. Our results showed that for our datasets, unpruned M5$'$ far outperforms pruned M5$'$. Unpruned M5$'$ produces significantly lower errors in 23 datasets, while pruned M5$'$ gives a lower error on only 1 dataset. However, to study different bias and variance settings, we present results on both pruned and unpruned M5$'$.
In the following subsections we summarize the key comparisons from Table 3.

IB vs. SGB: Our results show that IB significantly outperforms SGB, both in terms of win/draw/loss records and error reduction. The differences in performance are more dramatic on pruned M5$'$, where IB performs better than SGB on 23 of the 25 datasets, and produces twice the error reduction on average. The marked performance difference on pruned M5$'$ can be attributed to IB's superior bias-reduction.

	IB	BagSGB	BagGB	SGB	GB	Bag	M5′	%ErrRed
IB	-	13/1/11	10/6/9	10/12/3	18/4/3	16/1/8	17/1/7	16.44
BagSGB	11/1/13	-	10/8/7	16/5/4	15/5/5	18/4/3	21/2/2	16.35
BagGB	9/6/10	7/8/10	-	10/11/4	15/5/5	16/2/7	18/2/5	15.61
SGB	3/12/10	4/5/16	4/11/10	-	12/10/3	13/1/11	14/5/6	14.39
GB	3/4/18	5/5/15	5/5/15	3/10/12	-	13/0/12	13/1/11	7.45
Bag	8/1/16	3/4/18	7/2/16	11/1/13	12/0/13	-	17/5/3	1.98
M5′	7/1/17	2/2/21	5/2/18	6/5/14	11/1/13	3/5/17	-	-

(a)Base learner: unpruned M5′

	IB	BagSGB	BagGB	SGB	GB	Bag	M5′	%ErrRed
IB	-	18/4/3	20/3/2	23/1/1	24/0/1	19/3/3	22/1/2	16.89
BagSGB	3/4/18	-	9/9/7	19/6/0	18/6/1	19/4/2	22/3/0	11.82
BagGB	2/3/20	7/9/9	-	18/7/0	17/7/1	21/2/2	22/2/1	11.85
SGB	1/1/23	0/6/19	0/7/18	-	2/18/5	13/5/7	16/8/1	8.14
GB	1/0/24	1/6/18	1/7/17	5/18/2	-	13/4/8	16/7/2	8.55
Bag	3/3/19	2/4/19	2/2/21	7/5/13	8/4/13	-	16/7/2	2.59
M5′	2/1/22	0/3/22	1/2/22	1/8/16	2/7/16	2/7/16	-	-

(b)Base learner: pruned M5′

Table 3. Win/draw/loss records of algorithms in rows compared with algorithms in columns. The last column presents the average percentage reduction of error of the algorithm in the row compared with using M5′.

SGB, BagGB and BagSGB: BagGB performs significantly better than SGB, both for pruned and unpruned M5′. Similarly to IB, the differences are more pronounced on pruned M5′, where BagGB wins over SGB on 18 of the datasets, with no significant losses. The results suggest that applying bootstrap sampling to GB has a better variance-reducing effect than the randomization incorporated in SGB. In fact, applying bagging to SGB (BagSGB), can significantly drive down the error of SGB, as can be seen for both M5′ settings. BagSGB performs marginally better than BagGB in terms of win/draw/loss records, though their error reductions are quite comparable.

IB vs. BagGB/BagSGB: On unpruned M5′, BagGB and BagSGB perform comparably to IB both in terms of win/draw/loss records and error reduction — all methods producing approximately a 16% reduction in root-mean-squared error. However, for pruned M5′ trees, which have higher bias, IB exhibits a significant advantage over BagGB and BagSGB. In this case, it wins over BagGB and BagSGB on 20 and 18 datasets respectively. We also observe approximately a 5% difference in error reduction between IB and the other methods. IB's effectiveness at debiasing learners makes it a clear winner in higher bias settings.

SGB vs GB: Our results on unpruned M5′, which has high variance error, support the claim in [7] that SGB has a better variance-reducing effect than

	Friedman1			Friedman2			Friedman3($\times 10^{-4}$)		
	Bias^2	Var.	Err.	Bias^2	Var.	Err.	Bias^2	Var.	Err.
IB	0.94	0.65	1.59	126	127	253	2.9	1.2	4.2
BagSGB	1.49	0.48	1.96	237	149	386	4.4	1.0	5.5
BagGB	1.44	0.55	1.98	171	279	450	3.8	1.2	5.0
SGB	1.04	1.90	2.94	201	582	784	3.3	2.2	5.5
GB	0.97	1.31	2.28	136	322	458	2.9	2.4	5.3
Bag	4.80	0.43	5.24	1087	144	1230	9.9	0.8	0.7
M5$'$	4.69	1.09	5.79	1056	398	1454	9.4	2.0	1.4

Table 4. Bias-variance decompositions of mean squared error on synthetic data.

GB. SGB on average reduced 14.39% of the error of unpruned M5$'$, while GB reduced only 7.45%. However, SGB has significant wins in only 12 datasets and ties with GB in 10. Although, the error reduction of SGB is quite good, the win/draw/loss results do not suggest as significant an advantage of SGB over GB as in [9]. In fact, on pruned M5$'$, the performance of SGB and GB are tied on 18 datasets, with SGB performing slightly worse on the other datasets.

Bias-variance reduction vs. bias or variance reduction: GB and bagging focus solely on reducing the bias and variance of learners respectively. On the other hand, IB, SGB, BagGB and BagSGB attempt to reduce both the contribution of bias and variance to error. For brevity, we will refer to these four methods as BV-methods. Our results show that in general, the BV-methods have a significant advantage over GB and bagging, even when using the same number of base models. When compared to GB, BV-methods perform significantly better on at least 12 datasets and lose on at most 5 datasets. The only exception is SGB using pruned M5$'$, which loses to GB by a margin of 3 datasets. Even when compared to bagging, SGB is less effective than the other BV-methods. It wins by a margin of 2 (13 wins vs. 11 losses) and 6 (13 wins vs. 7 losses) when using unpruned M5$'$ and pruned M5$'$ respectively. The other BV-methods win by at least 16 datasets and lose on at most 8 datasets when compared to bagging. The results clearly indicate that combining techniques for bias and variance reduction is more effective than focusing on either bias or variance alone.

5 Bias-Variance Analysis

We explain most of our results based on how the different learners effect the bias and variance components of the error. To support our conjectures, we ran additional experiments to explicitly measure the bias and variance reducing effects of the methods presented. We performed BV decompositions on three synthetic datasets, as done in [15]. We do not introduce noise in these datasets, so that the evaluation of the bias and variance reduction capability of a learner is not

Friedman1			Friedman2			Friedman3		
Bias^2	Var.	Err.	Bias^2	Var.	Err.	Bias^2	Var.	Err.
IB	Bag	IB	IB	IB	IB	IB	Bag	IB
GB	BagSGB	BagSGB	GB	Bag	BagSGB	GB	BagSGB	BagGB
SGB	BagGB	BagGB	BagGB	BagSGB	BagGB	SGB	IB	GB
BagGB	IB	GB	SGB	BagGB	GB	BagGB	BagGB	BagSGB
BagSGB	M5$'$	SGB	BagSGB	GB	SGB	BagSGB	M5$'$	SGB
M5$'$	GB	Bag	M5$'$	M5$'$	Bag	M5$'$	SGB	Bag
Bag	SGB	M5$'$	Bag	SGB	M5$'$	Bag	GB	M5$'$

Table 5. Methods in order of increasing bias, variance and overall error.

confounded with its ability to handle noise. The target functions for the three datasets is given below:

Friedman1: $y = 10sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$
Friedman2: $y = (x_1^2 + (x_2 x_3 - (1/x_2 x_4))^2)^{1/2}$
Friedman3: $y = tan^{-1}(\frac{x_2 x_3 - (1/x_2 x_4)}{x_1})$

The first dataset, *Friedman1*, has 10 input variables $x_1, ..., x_{10}$, which are uniformly distributed over $[0, 1]$. The other two datasets, *Friedman2* and *Friedman3*, have 4 inputs x_1, x_2, x_3, x_4, which are uniformly distributed over the ranges:

$$0 \leq x_1 \leq 100$$
$$20 \leq (x_2/2\pi) \leq 280$$
$$0 \leq x_3 \leq 1$$
$$1 \leq x_4 \leq 11$$

To estimate bias and variance we used the method proposed by Kohavi and Wolpert [16], appropriately modified for regression (as opposed to classification). Each dataset was divided into two halves, D and E. D was used to draw our sample of training sets from, and E was used to estimate the terms in the BV decomposition. We generated 50 training sets from D sampled uniformly at random without replacement. Each training set of size 200 was selected from the pool of 400 examples in D. Each learning algorithm was run on each of the training sets and the squared bias and variance terms were calculated on set E based on equation 1. These values were averaged over all 50 train-test cycles.

Table 4 presents the overall error, bias^2 and variance terms for all algorithms applied to unpruned M5$'$ trees. The results for pruned trees were qualitatively similar, though in general the errors were higher for all methods. For SGB and BagSGB, we used a fraction $f = 0.6$; which is roughly equivalent to drawing bootstrap samples at each iteration. Table 5 presents the different algorithms in the order of increasing bias, variance and overall error on each dataset.

We observe that GB performs very well at reducing bias, but does not perform well at variance reduction. In fact, on *Friedman1* and *Friedman3*, GB actually increases the variance of the base learner. Analogously, bagging can increase the bias of the learner, but performs very well in terms of variance reduction. By combining the power of bagging and GB, BagGB produces a lower overall error than each of its components. Similarly, BagSGB improves on the bias reduction of bagging and the variance reduction of SGB, and as a result produces a lower overall error than both component algorithms. IB shows the best performance on overall error, and it appears to be quite effective in reducing both bias and variance. In fact, IB also performs the best in terms of bias reduction on all three datasets.

6 Future Work and Conclusion

We compared four approaches to combining bias and variance reduction techniques — Stochastic Gradient Boosting, Iterated Bagging, bagging Gradient Boosting and bagging Stochastic Gradient Boosting. Our results demonstrate that methods for combining bias and variance reduction (BV-methods) are more effective than methods that focus either on bias or variance in isolation. We also showed that while SGB often improves on GB, it is not very consistent and is easily outperformed by the other BV-methods. Experimental results show that for unpruned trees, which are low-bias learners, BagGB and BagSGB perform somewhat comparably to IB. However, IB, being a more effective bias-reduction method, performs much better compared to other algorithms when applied to pruned trees.

In our study, we restricted our methods to building at most 100 models each. Typically, the performance of these ensemble methods (or additive models) improve with ensemble size. In future work, we would like to explore the relationship between the number of models used and the effectiveness of each method. Experimenting with base learners other than decision trees, such as neural networks and support vector machines would also be very useful.

7 Acknowledgements

We would like to thank Horris Tse for assistance with running some of the experiments. Prem Melville and Raymond Mooney were supported by DARPA grant HR0011-04-1-007.

References

[1] Geman, S., Bienenstock, E., Dorsat, R.: Neural networks and the bias/variance dilemma. Neural Computation **4** (1992) 1–58

[2] Friedman, J.: Greedy function approximation: a gradient boosting machine. Technical report, Stanford University Statistics Department (1999)

[3] Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140

[4] Valentini, G., Dietterich, T.G.: Low bias bagged support vector machines. In: Proceedings of 20th International Conference on Machine Learning (ICML-2003), Washington, DC (2003) 752–759

[5] Webb, G.: Multiboosting: A technique for combining boosting and wagging. Machine Learning **40** (2000) 159–196

[6] Breiman, L.: Using iterated bagging to debias regressions. Machine Learning **45** (2001) 261–277

[7] Friedman, J.: Stochastic gradient boosting. Technical report, Stanford University Statistics Department (1999)

[8] Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Verlag, New York (2001)

[9] Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Technical report, Stanford University Statistics Department (2000)

[10] Breiman, L.: Using adaptive bagging to debias regressions. Technical report, UC Berkeley Statistics Department (1999)

[11] Blake, C.L., Merz, C.J.: UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html (1998)

[12] Wang, Y., Witten, I.: Inducing model trees for continuous classes. ECML Poster Papers (1997) 128–137

[13] Quinlan, J.: Learning with continuous classes. In: Proceedings of 5th Australian Joint Conference on Artificial Intelligience. (1992) 343–348

[14] Breiman, L., Friedman, J.H., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth and Brooks, Monterey, CA (1984)

[15] Friedman, J.H.: Multivariate adaptive regression splines. The Annals of Statistics **19** (1991) 1–141

[16] Kohavi, R., Wolpert, D.H.: Bias plus variance decomposition for zero-one loss functions. In Saitta, L., ed.: Proceedings of the Thirteenth International Conference on Machine Learning (ICML-96), Morgan Kaufmann (1996)