Senior Thesis in Mathematics

Integrating Random Forests into the Bag of Little Bootstraps

Author:
Yenny Zhang

Advisor:
Dr. Johanna Hardin

Submitted to Pomona College in Partial Fulfillment
of the Degree of Bachelor of Arts

April 27, 2017

Abstract

The widespread use of computationally intensive statistical procedures has inspired a world of research and practice that depends on massive, high-dimensional datasets, and the need to efficiently process them. Bootstrapping, the act of resampling with replacement from one sample, is one popular computational method, but it sacrifices time and memory efficiency with large sample sizes. The "Bag of Little Bootstraps" (BLB) is a variation of bootstrapping that avoids this loss of time and memory efficiency by introducing a layer of subsampling before bootstrapping. The original creators of BLB proved its statistical effectiveness using the mean as an estimator, and here, we examine the effects of using random forests, instead.

Contents

Chapter 1

Introduction

Bootstrapping and other resampling methods are invaluable tools for understanding sampling distributions in modern statistics. Given some parameter we would like to estimate, or some model we would like to fit, bootstrapping is the act of repeatedly drawing resamples from one sample of data and refitting or remeasuring an estimator on each resample [James et al., 2014, pages 175]. Though simple-sounding in nature, this powerful tool gives us access to more information about our model or estimator than allowed with the original sample.

Bootstrapping is applicable to many situations, but the most familiar application of bootstrapping is to assess the standard error of an estimator. The main idea of bootstrapping is to treat a sample from the population of interest as an empirical distribution of the data. Then, the sampling distribution of that statistic can be estimated by bootstrapping from the sample. If n is the size of the original sample, in bootstrapping, one draws r resamples with replacement of size n from the original sample. Then, the statistic of interest is computed from each resample, and all r statistics are aggregated to form a bootstrap distribution which approximates the theoretical sampling distribution of the statistic.

Although incredibly useful, bootstrapping has its limitations. The accuracy of estimating a sampling distribution is dependent upon the size of the original sample, n, and the number of resamples, r. As both r and n become large, the bootstrap distribution will come closer to approximating the shape and spread of true sampling distribution. But if n and r are too large,

then the computation required to perform bootstrapping is often not fast nor space efficient enough to justify its usage. Computational issues have become increasingly relevant due to the current trend in "big data"; massive, high-dimensional datasets are now ubiquitous. Some methods have been proposed to combat the computational problem; subsampling [Politis et al., 1999] and the m out of n bootstrap [P.J. Bickel and van Zwet, 1997] are two examples, but a recent procedure called the Bag of Little Bootstraps (BLB) has been shown to "yield a robust, computationally efficient means of assessing the quality of estimators" [Kleiner et al., 2014, pages 1–2]. BLB combines subsampling and bootstrapping such that bootstrapping is performed on small, independent subsets from one sample. Because bootstrapping is only performed on each subset, the computational cost is favorably rescaled to the size of each subset. As a result, this structure is well-suited to parallel computing architectures that are often used to process large datasets by splitting a process into parts that will run simultaneously on different processors. BLB was introduced in the scenario to estimate the mean, but because bootstrapping is such a flexible and accommodating resampling method for other statistics and models, it is natural to ask if BLB can be applied to more complex models.

Random forests are one extremely popular ensemble learning method used for classification and regression on large datasets. However, unlike the mean, there is no single parameter to estimate; the output of a random forest depends on a test observation whose response is to be predicted. Thus it is not a trivial problem to substitute the mean in the BLB structure with random forests. Random forests are a common tool used to find accurate predictions for a set of test observations. Thus, we aim to see if we can quantify the accuracy of a random forest prediction from BLB. We will explore how to integrate random forests into the BLB structure of resampling, and if doing so is a robust method in maintaining the results of performing traditional bootstrapping with random forests.

Chapter 2

Bag of Little Bootstraps

The Bag of Little Bootstraps is a relatively new procedure that combines the features of bootstrapping and subsampling into a resampling method that can accompany massive data sets, while still maintaining the same power of bootstrapping [Kleiner et al., 2014]. This section will describe the Bag of Little Bootstraps' notation, procedure, and its statistical performance.

2.1 Setting and Notation

Assume that we observe a sample $X_1, \ldots, X_n$ [1] of training data that is drawn from some unknown population distribution P . We denote its corresponding empirical distribution as $\mathbb{P}_n = \frac{1}{n}\sum_{i=1}^{n} \delta_{X_i}$. Using only the observed data we compute some estimate $\hat{\theta}_n \in \Theta$ of some population value $\theta \in \Theta$ that is associated with the population distribution of P. When we want to specify that we are using data, for example, $\mathbb{P}_n$, to calculate our estimator $\hat{\theta}$, we shall specify $\hat{\theta}_n = \hat{\theta}(\mathbb{P}_n)$. Here we will choose $\hat{\theta}_n$ to be the sample mean.

We denote the true sampling distribution of $\hat{\theta}_n$, as $Q_n(P)$. Recall that bootstrapping aims to estimate the sampling distribution $Q_n(P)$, from which we can compute an estimator quality assessment $\xi(Q_n(P), P)$, which could be a confidence interval, standard error, or bias. For simplicity, we will use $\xi(Q_n(P))$ instead of $\xi(Q_n(P), P)$, to represent a confidence interval for a given parameter. Then it follows that the bootstrap computes the plug-in approximation $\xi(Q_n(P)) \approx \xi(Q_n(\mathbb{P}_n))$. The bootstrap algorithm repeatedly

[1] All notation is taken from [Kleiner et al., 2014].

takes resamples with replacement of size n from $\mathbb{P}_n$, computes $\hat{\theta}_n$ on each resample, and forms the empirical distribution $\mathbb{Q}_n^*$, with the end goal of approximating $\xi(Q_n(P)) \approx \xi(\mathbb{Q}_n^*)$.

2.2 Bag of Little Bootstraps

2.2.1 Description

The Bag of Little Bootstraps (BLB) is unique in that it employs bootstrapping on mutually exclusive small subsets of our original observed sample $X_1, \ldots, X_n$ such that each subset has its own bootstrap distribution.

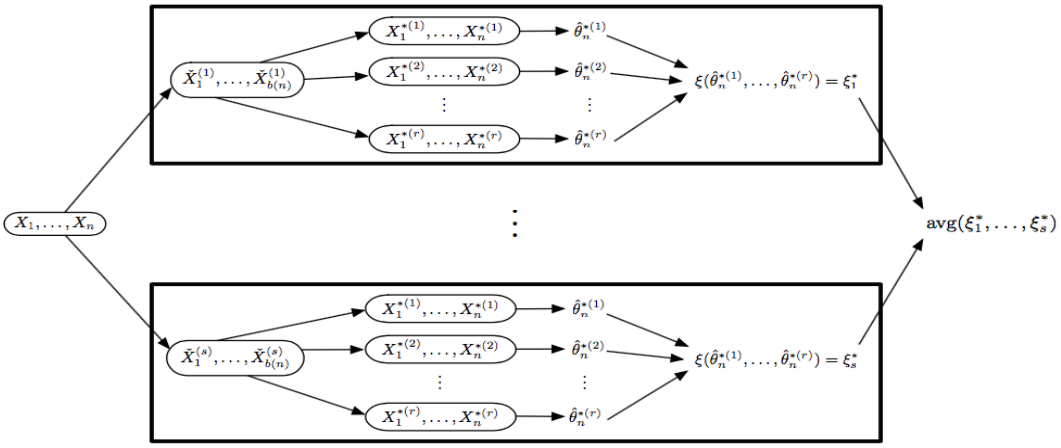

Figure 2.1: The Bag of Little Bootstraps.

Let each subset from our observed set be of size $b < n$. BLB randomly partitions the n points into s subsets of size b. Let $\mathcal{I}_1, \ldots, \mathcal{I}_s \subset \{1, \ldots, n\}$ be the randomly sampled subsets, such that $|\mathcal{I}_j| = b, \forall j$, and let $\mathbb{P}_{n,b}^{(j)} = \frac{1}{b}\sum_{i \in \mathcal{I}_j} \delta_{X_i}$ be the empirical distribution that corresponds to the jth subset. Then, BLB's estimate of $\xi(Q_n(P))$ is

$$\frac{1}{s}\sum_{j=1}^{s} \xi(Q_n(\mathbb{P}_{n,b}^{(j)})). \tag{2.1}$$

Because we cannot compute $\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$ analytically, we can approximate $\xi(Q_n(\mathbb{P}_{n,b}^{(j)})) \approx \xi(\mathbb{Q}_{n,j}^*)$ by bootstrapping on each jth subset. That is, we repeatedly resample n points from $\mathbb{P}_{n,b}^{(j)}$, the empirical distribution corresponding to the jth subset, to form the empirical distribution $\mathbb{P}_{n,b}^{(j)}$ of our computed estimates of $\hat{\theta}_n$. We will see how to resample n points from $b < n$ points in subsection 2.2.3.

Symbol	Meaning
$X_1, \ldots, X_n$	original sample
$\mathcal{I}_1, \ldots, \mathcal{I}_s$	subsets of original sample $X_1, \ldots, X_n$
$\mathbb{P}_n = \frac{1}{n}\sum_{i=1}^n X_i$	empirical distribution of $X_1, \ldots, X_n$
$\mathbb{P}_{n,b}^{(j)} = \frac{1}{b}\sum_{i \in \mathcal{I}_j} \delta_{X_i}$	empirical distribution of the jth subset
$\mathbb{P}_{n,k}^* = \frac{1}{n}\sum_{a=1}^b n_a \delta_{X_i a}$	empirical distribution of the kth resample (for some subset)
$\hat{\theta}$	estimator of interest
$\hat{\theta}_{n,k}^* \leftarrow \hat{\theta}(\mathbb{P}_{n,k}^*)$	estimator computed from the kth resample (for some subset)
$Q_n(P)$	true sampling distribution of $\hat{\theta}$
$\mathbb{Q}_{n,j}^* \leftarrow \frac{1}{r}\sum_{k=1}^r \delta_{\hat{\theta}_{n,k}^*}$	the empirical distribution of bootstrapped estimates from the jth subset's bootstrap distribution
$\xi(Q_n(P))$	confidence interval associated with the sampling distribution $Q_n(P)$
$\xi(\mathbb{Q}_{n,j}^*)$	confidence interval estimated from the jth subset
$\hat{\xi}(Q_n(P)) \leftarrow \frac{1}{s}\sum_{j=1}^s \xi(\mathbb{Q}_{n,j}^*)$	estimate of $\xi(Q_n(P))$

Table 2.1: Table of notation used in BLB.

Algorithm 1 Bag of Little Bootstraps (BLB)

Input: $X_1, \ldots, X_n$, data
$\hat{\theta}$, estimator of interest
b, subset size
s, number of sampled subsets
r, number of resamples
ξ, estimator quality assessment

Output: $\hat{\xi}(Q_n(P))$, confidence interval estimate

for $j = 1 \ldots s$ **do**
`// Subsample the data`
Randomly sample a set $\mathcal{I} = \{i_1, \ldots, i_b\}$ of b indices from $\{1, \ldots, n\}$ without replacement
for $k = 1 \ldots r$ **do**
`// Resample the data`
Sample up to size n: Sample $(M_1, \ldots, M_b) \sim \text{Multinomial}(n, \frac{1}{b})$
$\mathbb{P}^*_{n,k} \leftarrow \frac{1}{n} \sum_{a=1}^{b} n_a \delta_{X_{ia}}$
$\hat{\theta}^*_{n,k} \leftarrow \hat{\theta}(\mathbb{P}^*_{n,k})$
end for
$\mathbb{Q}^*_{n,j} \leftarrow \frac{1}{r} \sum_{k=1}^{r} \delta_{\hat{\theta}^*_{n,k}}$
end for

return
$\hat{\xi}(Q_n(P)) \leftarrow \frac{1}{s} \sum_{j=1}^{s} \xi(\mathbb{Q}^*_{n,j})$

Figure 2.2: BLB Algorithm.

2.2.2 Multinomial Method

Note that in each of the s subsets, we bootstrap by taking a resample of size n from the subset of size $b < n$. We are able to bootstrap from size $b < n$ up to n by using a multinomial random variable $\mathbf{M} = (M_1, M_2, \ldots, M_b)$ that helps "imitate" true bootstrapping. Each M_i coefficient represents the number of times we encounter the ith observation from the subset in the resample. By assigning an integer coefficient to every observation, space does not have to be allocated in memory for n separate data points, but for b data points and b coefficients. Each multinomial random variable is generated with a positive integer, n, which is the size of the resample, and $\mathbf{p} = (p_1, p_2, \ldots, p_b)$, a vector of probabilities. When bootstrapping from a subset in BLB, $\mathbf{p}$ will always be a b-dimensional vector where $p_1 = p_2 = \ldots = p_b = \frac{1}{b}$ because each observation has a probability of $\frac{1}{b}$ of being drawn. This aspect of BLB avoids the traditional bootstrap's "problematic need for repeated computation of the estimate on resamples having size comparable to that of the original dataset" [Kleiner et al., 2014].

Figure 2.2 shows an example of resampling from a subset of size $b = 5$ up to size $n = 100$. In this example, $\mathbf{M} \sim Mult(100, (\frac{1}{b}, \frac{1}{b}, \frac{1}{b}, \frac{1}{b}, \frac{1}{b}))$, and the vector of coefficients is $\mathbf{M} = (21, 19, 23, 22, 15)$. Note that this is simply one permutation of multinomial coefficients for this resample with the specified parameters of $b = 5, n = 100$.

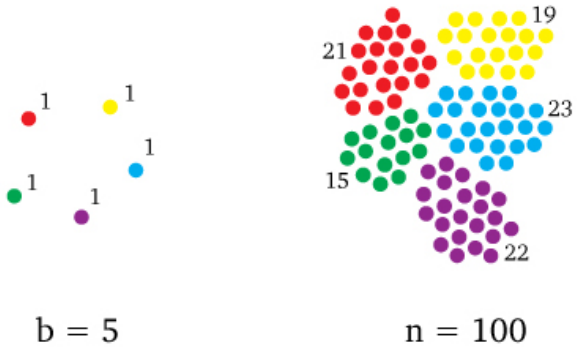

Figure 2.3: Resamping from size $b = 5$ up to size $n = 100$.

2.3 Statistical Performance

Our main premise in using BLB is that it maintains the same statistical properties of standard bootstrapping. Here we provide evidence from Kleiner et al. that BLB has both asymptotic consistency and higher-order correctness, the two main statistical properties of the bootstrap.

2.3.1 Consistency

We would like to show that the estimate of standard error $\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$ converges in probability to the true standard error $\xi(Q_n(P))$ as $n, b \to \infty$.

Theorem 2.1. *[Kleiner et al., 2014].*
We assert that the following is true as $n \to \infty$, for any sequence $b \to \infty$, and for any fixed s.

$$\frac{1}{s}\sum_{j=1}^{s}\xi(Q_n(\mathbb{P}_{n,b}^{(j)})) - \xi(Q_n(P)) \xrightarrow{P} 0 \tag{2.2}$$

That is, the BLB estimate $\frac{1}{s}\sum_{j=1}^{s}\xi(Q_n(\mathbb{P}_{n,b}^{(j)}))$ approaches the population value $\xi(Q_n(P))$ in probability.

2.3.2 Higher-order Correctness

For BLB to be higher-order correct, it needs to converge to the true value of $\xi(Q_n(P))$ at a rage of $O_P(\frac{1}{n})$ or faster if s and b grow at a sufficient rate.

Theorem 2.2. *[Kleiner et al., 2014].*
Under certain regularity conditions, we assert that the following is true as $n \to \infty$, for any sequence $b \to \infty$, and for any fixed s.

$$\left|\frac{1}{s}\sum_{j=1}^{s}\xi(Q_n(\mathbb{P}_{n,b}^{(j)})) - \xi(Q_n(P))\right| = O_P\left(\frac{1}{n}\right), \tag{2.3}$$

in which case BLB enjoys the same level of asymptotic higher-order correctness as the bootstrap.

2.4 Conclusion

The Bag of Little Bootstraps is a relatively new alternative to bootstrapping, which is the traditional resampling method that is used to estimate the theoretical sampling distribution of an estimator. In addition to sharing the statistical properties of traditional bootstrapping, BLB is also well-suited to large-scale datasets. In BLB, although bootstrapping is performed on small subsets of the original sample, each bootstrap resample is scaled to to the size of the original sample by using the multinomial method. This gives BLB a "better computational profile" [Kleiner et al., 2014], such that memory and computation are markedly less than that of traditional bootstrapping when n is extremely large. Because BLB has the same statistical properties of the bootstrap, it was presented in the scenario to estimate the mean. In the following sections, we will introduce random forests and how BLB can be applied to estimate the variability of the random forest model.

Chapter 3

Random Forests

Whereas the mean is a function of data that is concerned with estimating a fixed population parameter, μ, a random forest is a model that seeks to predict the categorical or numerical response of a new test observation, or set of observations, given by the model. More importantly, random forests are built off of bootstrapped decision trees that are constructed through the 'classification and regression tree' (CART) algorithm. Decision trees and random forests are predictive models for both classification and regression problems, but here we will only focus on the regression case. We will define the building blocks of a random forest, the CART algorithm that is used to build decision trees, and how random forests are built from those decision trees.

3.1 Classification and Regression Trees

Tree-based methods can be used for classification and regression problems by partitioning a data-space into a finite number of regions that represent the most homogenous space with respect to the response variable for the data. A tree nicely summarizes the splitting rules that determine the best regions. Random forests are built from *binary decision trees*, which consist of repeated splits of a dataset into two descendant subsets. Each subset of data is contained in a *node*. In each split of a binary decision tree, we refer to the two descendent nodes as *daughter nodes*, and the node from which they came the *parent node*. More importantly, each node in a binary decision tree is either an *internal node* or a *terminal node*. The terminal nodes of a tree

define subsets that provide the best final partitioning of our data according to a sums of squares criterion.

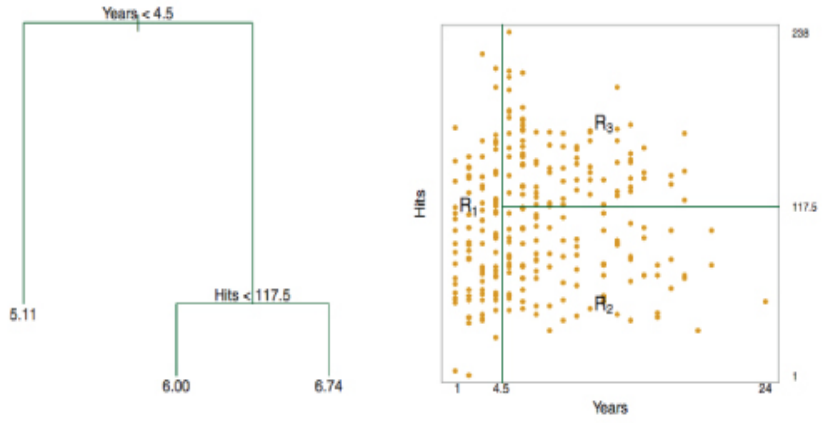

Figure 3.1: A simple decision tree built to predict salary from baseball player characteristics [James et al., 2014, pages 304-305].

Each node is defined by an "if-else" condition such that a subset of observations fit within that node only if they satisfy the condition. One of the most popular methods for tree-based regression and classification problems, called CART (Classification and Regression Trees), builds a decision tree by recursively partitioning data into two parts to create a model that can predict a new observation's response value given a set of known explanatory characteristics.

3.1.1 CART Overview

There are two major steps to the CART algorithm: the first is to recursively build a single tree on the data, otherwise known as our *training sample*. The second part consists of "pruning" the tree. The CART algorithm builds a tree by finding the "best" binary split at each node, but the algorithm is slightly different for classification problems than for regression problems.

3.1.2 Finding the Best Binary Split

The central operation in the CART algorithm is to make a split in the explanatory variable space. There are many several possible ways to make a binary split in the training sample X. Each possible split is defined by an explanatory variable X_j and a cutpoint s such that the explanatory variable space can be partitioned into 2 regions, R_1 and R_2.

If X_j is discrete, let s_0 be the set of values for X_j defined by the cutpoint s. Then R_1 and R_2 are defined as the following:

$$\begin{aligned} R_1 :&\{X|X_j \in s_0\} \\ R_2 :&\{X|X_j \notin s_0\} \end{aligned} \tag{3.1}$$

For example, suppose X_j is an explanatory variable for color. Possible values for X_j are blue, green, fuchsia, and red. One possible cutpoint s could divide X_j into regions $R_1 : \{X|X_j \in \{blue, green\}\}$ and its complement, $R_2 : \{X|X_j \notin \{blue, green\}\}$.

If X_j is continuous, R_1 and R_2 are defined as the following:

$$\begin{aligned} R_1 :&\{X|X_j < s\} \\ R_2 :&\{X|X_j \geq s\} \end{aligned} \tag{3.2}$$

For example, suppose X_j is an explanatory variable for salary, whose range is from $\$10,000$ to $\$500,000$. One possible cutpoint $s = \$129,000$ could divide X_j into regions $R_1 : \{X|X_j < \$129,000\}$ and its complement, $R_2 : \{X|X_j \geq \$129,000\}$.

3.1.3 Growing a Classification Tree

In classification problems, the main goal is to predict a categorical response variable based off of numeric or categorical predictors. Given a training set $\mathcal{L}$, let us define the different values of our categorical response variable as *classes*, and our explanatory variables as *predictors*. Our set of observations fall into the set of classes $C = \{1, 2, \ldots, K\}$, with the set of possible predictors $P = \{X_1, X_2, \ldots, X_p\}$. If the size of $\mathcal{L}$ is n, each ith observation can be represented as $(y_i, \boldsymbol{x_i})$, where $\boldsymbol{x_i}$ is a vector $(x_{i1}, x_{i2}, \ldots, x_{ip})$ and y_i is categorical.

For a classification problem, the CART algorithm determines the best split of each parent node such that the class distributions of each of its daughter nodes are "purer" than their parent nodes'. In other words, we seek to minimize a measure of "impurity" with each iterative split.

Impurity

Impurity can be measured in multiple ways, but some common properties hold for each measure of impurity. Let $p_{k,t}$ be the proportion of observations in class k at node t. Impurity is a function ϕ defined on the set of all K-tuples of $(p_{1,t}, \ldots, p_{K,t})$ satisfying $k = 1, 2, \ldots, K$, and $\sum_{k=1}^{K} p_{k,t} = 1$.
For any tree, we can define the impurity $I(t)$ at each node t as:

$$I(t) = \phi(p_{1,t}, p_{2,t}, \ldots, p_{k,t}) \tag{3.3}$$

Some common measures of impurity include the *Gini index*, *Cross-entropy*, and *Misclassification error*. For the tree-building portion of the CART algorithm, we focus on the *Gini index*.

The Gini Index

We define the *Gini Index* at node t:

$$G(t) = \sum_{k=1}^{K} p_{k,t}(1 - p_{k,t}), \tag{3.4}$$

which we can rewrite as:

$$G(t) = 1 - \sum_{k=1}^{K} {p_{k,t}}^2. \tag{3.5}$$

Because impurity is a measure of how homogenous a node is, a node whose observations all fall into one class would be considered homogenous, or pure and result in $G(t) = 0$. A node whose observations are distributed evenly across all possible classes has maximum impurity. There is a maximum value of $G(t)$ for each k.

Using the Gini index, we can define the impurity at node t:

$$I(t) = \sum_{k=1}^{K} p_{k,t} \cdot G(t). \tag{3.6}$$

Each binary split can be across either a discrete or continuous explanatory variable, as described in section 3.1.2. Once a split is made in some parent node$t$, region R_1 describes the data that falls into the left daughter node t_L and R_2 describes the data that falls into right daughter node t_R. It follows that for each binary split, we can calculate the overall reduction in impurity as a difference of the impurity of a parent node t and the sum of the impurities of both t_L, and t_R:

$$\Delta I = I(t) - (I(t_L) + I(t_R)). \tag{3.7}$$

Because we are building a tree that predicts a qualitative response, a binary split is just assigning qualitative values to the left daughter node, and the remaining qualitative values to the right daughter node. The CART algorithm is greedy; it will find the best split by considering all possible splits on a node across all explanatory variables, and choosing the split that maximizes equation 3.7. The algorithm will iterate recursively on each of the resulting daughter nodes until a stopping condition is reached. The resulting tree is one whose terminal nodes are reasonably pure.

3.1.4 Growing a Regression Tree

In a regression problem, the main goal is to predict a numeric response variable based off of numeric or categorical predictors. Our criteria for splitting is to minimize sum of squared errors of the response.

At each node t where N_t is the number of observations in node t, we can calculate an arithmetic mean of responses contained in node t:

$$\bar{y_t} = \frac{1}{N_t} \sum_{n=1}^{N_t} y_n. \tag{3.8}$$

Using the mean response in each node, we can calculate the spread of responses within that node by finding the sum of squared differences from each

response. We define the following as the Sum of Squared Errors (SSE) for a given node t:

$$SSE_t = \sum_{n=1}^{N_t} (y_n - \bar{y_t})^2. \tag{3.9}$$

A high SSE value means that the responses in a node are spread farther apart from each other than if a node had a lower SSE. CART uses SSE to determine each split in the tree, or each partition of our training data, as described in section 3.1.2. The explanatory variable X_j and values that determine the best split are values that maximize the reduction in SSE. The algorithm will consider all predictors $X_1, \ldots, X_p$ and all possible values of s for each of the p predictors and calculate a reduction in SSE instead of impurity, which is the difference between parent node's SSE and the sum of the SSE for the daughter nodes:

$$\Delta SSE = SSE_t - (SSE_{t_L} + SSE_{t_R}). \tag{3.10}$$

The CART algorithm will iterate recursively to find the value the explanatory variable X_j and cutpoint s that maximize equation 3.10 on each of the resulting daughter nodes until a stopping condition is reached. The resulting tree is one whose terminal nodes contain values that are closely centered around the mean of their respective nodes.

3.1.5 Prediction

After partitioning the data and building a tree, there will be mutually exclusive regions $R_1, \ldots, R_j$ corresponding to the explanatory variables and one terminal node that corresponds to each of the j regions. Given a test observation, we can predict its response value by using its explanatory variables to determine which terminal node which gives the average response variable it falls into. Then, its predicted response value is the mean of the training observations in the region to which that test observation belongs according to the explanatory variables.

3.2 Bagged Learners

Decision trees often suffer from *high variance*, meaning if we were to split our training data into two parts at random, and fit a decision tree to both halves,

the results we get could be very different. This is related to the *instability* of decision trees, defined such that a small change in our training set will change the outcome of our best predictive tree model. Methods with *low variance* will produce similar predictive models if applied to different samples from the same population.

3.2.1 Bagging

One method, *bootstrap aggregating*, or *bagging*, is a procedure that reduces the variance of decision trees, resulting in a more accurate model. The basic idea behind bagging is to aggregate the results of multiple bootstrapped decision trees through averaging or majority vote.

Given a training dataset of size n $\mathcal{L}$ with observations $\{(y_i, \boldsymbol{x_i}), i = 1, 2, \ldots, n\}$, where y_i is either a categorical or numeric response which we want to predict, we can use the CART algorithm to create a decision tree $\varphi(\boldsymbol{x}, \mathcal{L})$ such that if the input is $\boldsymbol{x}$, we predict y using $\varphi(\boldsymbol{x}, \mathcal{L})$. Suppose we take B bootstrap resamples from $\mathcal{L}$ with replacement to make up the set $\{\mathcal{L}^{(b)}, b = 1, 2, \ldots, B\}$. Then, from each bootstrap resample $\mathcal{L}^{(b)}$, we can grow a decision tree such that we have a set of decision trees $\{\varphi(\boldsymbol{x}, \mathcal{L}^{(b)}), b = 1, 2, \ldots, B\}$. When we aggregate all our predictors, our bagged predictor for a regression tree is:

$$\varphi_B(\boldsymbol{x}) = \frac{1}{B} \sum_{b=1}^{B} \varphi(\boldsymbol{x}, \mathcal{L}^{(b)}), \tag{3.11}$$

and for a classification tree:

$$\varphi_B(\boldsymbol{x}) = \text{most common class}_{b \in B} \varphi(\boldsymbol{x}, \mathcal{L}^{(b)}). \tag{3.12}$$

By aggregating the predictions from highly variable decision trees, we can reduce the variability of new predictions from test data.

We can show why averaging reduces variance [Breiman, 1996]. Let $\phi(\boldsymbol{x}, \mathcal{L})$ be a predictor for the population $\mathcal{P}$ with independently drawn sample $\mathcal{L}$ with observations $\{(y, \boldsymbol{x})\}$, where y is numeric. Let $\phi_B(\boldsymbol{x})$ be our bagged predictor, where $\phi_B(\boldsymbol{x}) = \mathrm{E}_{\mathcal{L}}[\phi(\boldsymbol{x}, \mathcal{L})]$. The mean-squared-error for our predictor $\phi_B(\boldsymbol{x})$ is:

$$\begin{aligned} \mathrm{E}_{\mathcal{L}}[(y - \phi(\boldsymbol{x}, \mathcal{L}))^2] &= y^2 - 2y\mathrm{E}_{\mathcal{L}}[\phi(\boldsymbol{x}, \mathcal{L})] + \mathrm{E}_{\mathcal{L}}[\phi^2(\boldsymbol{x}, \mathcal{L})] \\ &= y^2 - 2y\phi_B(\boldsymbol{x}) + \mathrm{E}_{\mathcal{L}}[\phi^2(\boldsymbol{x}, \mathcal{L})]. \end{aligned} \tag{3.13}$$

When we apply the inequality $\mathrm{E}^2[X] \geq \mathrm{E}[X]^2$,

$$\begin{aligned} \mathrm{E}_{\mathcal{L}}[(y - \phi(\boldsymbol{x}, \mathcal{L}))^2] &= y^2 - 2y\phi_B(\boldsymbol{x}) + \mathrm{E}_{\mathcal{L}}[\phi^2(\boldsymbol{x}, \mathcal{L})] \\ &\geq y^2 - 2y\phi_B(\boldsymbol{x}) + \mathrm{E}_{\mathcal{L}}[\phi(\boldsymbol{x}, \mathcal{L})]^2 \\ &\geq y^2 - 2y\phi_B(\boldsymbol{x}) + \phi_B(\boldsymbol{x})^2 \\ &\geq (y - \phi_B(\boldsymbol{x}))^2. \end{aligned} \tag{3.14}$$

We can see that the mean-squared error of $\phi_B(\boldsymbol{x})$ is lower than the mean-squared error over averaged over $\mathcal{L}$ of $\phi(\boldsymbol{x}, \mathcal{L})$.

3.2.2 Random Forests

The *random forests* procedure [Breiman, 2001] is a modification of *bagging*. Its purpose is to build a large collection of de-correlated trees, and average their predictions. The difference between *random forests* and *bagging* is that random forests grow de-correlated trees by using bootstrapped data and by modifying one step of the tree-growing process. Before each split, given p predictor variables, only $m \leq p$ are randomly selected as candidates for splitting. Intuitively, reducing m will reduce the correlation between any pair of trees and hence reduce the variance of the average.

Consider the following example: if there is one very strong predictor in the data set, then CART will repeatedly prioritize choosing this variable as the splitting variable over the others, and all the bagged trees will be very similar to each other, and thus, highly correlated. By choosing a random subset of predictor variables to consider splitting on, usually $m = \sqrt{p}$, the resulting bagged trees will not be as similar to each other such that averaging them will lead to a overall larger reduction in variance than if we averaged decision trees based only on bootstrapping the data and not on subsetting the variables. The averaged trees of random forests are less variable than the average of standard bagged decision trees, making random forests a more viable option than bagged predictors.

Chapter 4

Integrating Random Forests and BLB

4.1 Motivation

Random forests are an extremely popular machine learning technique. However, in settings that involve building random forests on extremely large datasets, the computational burden of growing one random forest can be large enough to justify not using it. We propose that integrating random forests into the "Bag of Little Bootstraps" model can combat this computational burden, and still maintain the predictive appeal of traditional random forests. This section will explore the main differences between building traditional random forests on large datasets, and building random forests within the "Bag of Little Bootstraps" structure.

4.2 Differences

4.2.1 Runtime

Suppose we want to build a random forest on a training dataset of size $n = 1,000,000,000$ and $p = 50$ predictors. Recall that random forests are built off of aggregated decision trees, and a decision tree is built using the CART algorithm. For the case of building a regression decision tree, the heart of the CART algorithm lies in maximizing 4.1:

$$\Delta SSE = SSE_t - (SSE_{t_L} + SSE_{t_R}), \tag{4.1}$$

where t is the node to be split, t_L is a potential left daughter node and t_R is a potential right daughter node. Given some predictor variable X_j and a cutpoint s of X_j, all the observations for which $\{X|X_j < s\}$ is satisfied will fall into node t_L and the rest, those that satisfy $\{X|X_j \geq s\}$, will fall into node t_R. Recall that SSE is measured by:

$$SSE_t = \sum_{n=1}^{N_t} (y_n - \bar{y_t})^2, \tag{4.2}$$

where N_t is the number of observations in node t.

Then it follows that for some predictor X_j and cutpoint s, we can partition our predictor space into $R_1(j,s) = \{X|X_j < s\}$ and $R_2(j,s) = \{X|X_j \geq s\}$ which has the a corresponding reduction in SSE:

$$\Delta SSE = \sum_{n=1}^{N_t} (y_n - \bar{y_t})^2 - (\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{t_L})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{t_R})^2). \tag{4.3}$$

Because the CART algorithm needs to find the R_1 and R_2 (which correspond to nodes t_L and t_R, respectively) that maximize ΔSSE, it needs to look through all possible partitions of the data in node t. For a random forest, CART only considers $m \leq p$ predictor variables to split on. Therefore, determining the binary splits at each level of the tree means looking at m predictor variables for each node at each level. Note that for each predictor variable X_j, there are at most n possible cutpoints for which we need to calculate ΔSSE. Each level of partitioning would then need to check at most $m \cdot n$ partitions of the data to find determine t_L and t_R. The runtime would be on the order of $O(n)$, multiplied by the number of trees one wants to grow (the number of trees grown for a random forest is usually at least 500). If n is large, one can imagine that this process would take an extremely long time.

Fortunately, by integrating random forests into BLB, we can reduce the long runtime. We will delve into how random forests are built in the BLB structure in the next section, and then we will talk about what final predictions and estimates of standard error look like.

4.2.2 Growing Random Forests in BLB

Just as the original BLB structure computes $\hat{\theta}$ from each resample of a subsample, we propose that integrating random forests and BLB will similarly involve building a random forest from each resample of a subsample, as in figure below.

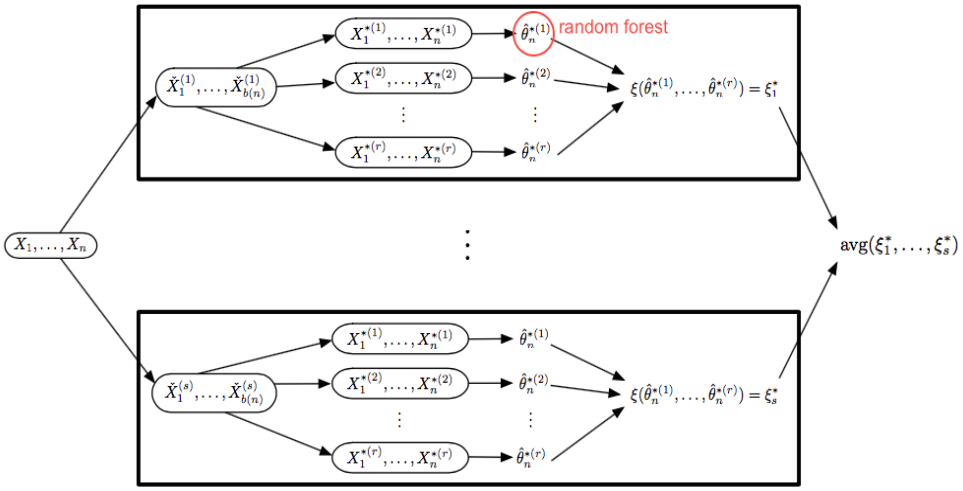

Figure 4.1: The Bag of Little Bootstraps with Random Forests.

Recall that in the Bag of Little Bootstraps, each resample of size n actually consists of b unique values where $b << n$. This is because a multinomial coefficient M_i accompanies each of the b unique values, where M_i is the count of each observation we see in the resample. Because there are only b unique values, each with a corresponding multinomial coefficient, the algorithm for growing a random forest on a resample in BLB is slightly different from that used for growing a random forest on a dataset with n unique values.

The main operation in the construction of a random forest is maximizing the sum of squared errors in equation 4.3 at each recursive partitioning of our data. Usually this operation is on the order of n, the size of our training dataset, as explained in the previous section.

However, in the Bag of Little Bootstraps, the number of observations in node t will always be less than or equal than b as opposed to always less than or equal to n. The algorithm will run on the order of $O(b)$, but with the variability associated with statistics calculated on samples of size n, not b. The multinomial coefficients, $\boldsymbol{M} = M_1, \ldots, M_b$ simply need to be integrated into equation 4.4 according to the following:

$$\Delta SSE = \sum_{n=1}^{N_t}(y_n - \bar{y}_t)^2 - (\sum_{i:x_i \in t_L} M_i \cdot (y_i - \hat{y}_{t_L})^2 + \sum_{i:x_i \in t_R} M_i \cdot (y_i - \hat{y}_{t_R})^2), \quad (4.4)$$

where $N_t \leq b$.

This works because the resample in BLB consists of repeats of each unique value from the subsample. Suppose we have some unique value from the sample, $(y_i, \boldsymbol{x_i})$ with its corresponding multinomial coefficient, M_i. Then, M_i is the number of times we see observation $(y_i, \boldsymbol{x_i})$ in the resample. Note that we only need to check the b unique values instead of n unique values. For example, in Figure 4.2, we only need to check splits between the 5 unique values, instead of within the repeats of each value.

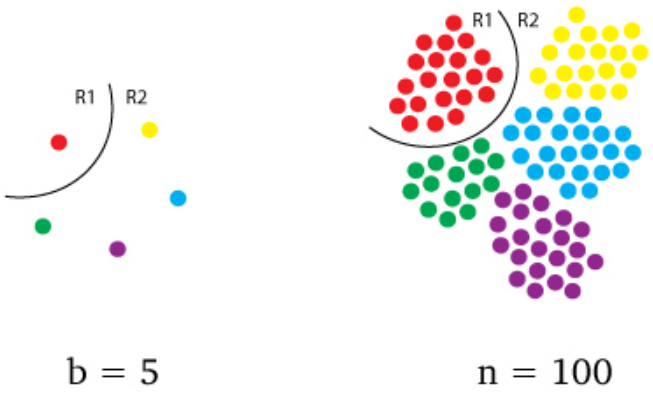

Figure 4.2: Splitting in BLB.

Here, we provide some pseudo-code to clarify the main modification to RF. Our new BLB-RF algorithm is provided on the next page.

Algorithm 2 Bag of Little Bootstraps (BLB) with Random Forests

Input: $X_1, \ldots, X_n$, training data
$Z_1, \ldots, Z_m$, test data
γ, a random forest
b, subset size
s, number of sampled subsets
r, number of resamples
ξ, estimator quality assessment

Output: $\bar{\boldsymbol{\theta}}^* = \bar{\theta}^*_1, \ldots, \bar{\theta}^*_m$, predicted values of test data
$\hat{\xi}(Q_n(P))$, confidence interval estimate

for $j = 1 \ldots s$ **do**
`// Subsample the data`
Randomly sample a set $\mathcal{I} = \{i_1, \ldots, i_b\}$ of b indices from $\{1, \ldots, n\}$ without replacement
for $k = 1 \ldots r$ **do**
`// Resample the data`
Sample up to size n: Sample $(M_1, \ldots, M_b) \sim \text{Multinomial}(n, \frac{1}{b})$
$\mathbb{P}^*_{n,k} \leftarrow \frac{1}{n} \sum_{a=1}^{b} n_a \delta_{X_{ia}}$
`// Build a random forest`
$\gamma_k \leftarrow \gamma(\mathbb{P}^*_{n,k})$
`// Get m predictions for test data`
$\hat{\boldsymbol{\theta}}^*_{\boldsymbol{n,k}} \leftarrow \gamma_k(Z_1, \ldots, Z_m)$
end for
$\mathbb{Q}^{(1)*}_{n,j}, \ldots, \mathbb{Q}^{(m)*}_{n,j} \leftarrow \frac{1}{r} \sum_{k=1}^{r} \boldsymbol{\delta}_{\hat{\boldsymbol{\theta}}^*_{\boldsymbol{n,k}}}$
end for
return
$\bar{\boldsymbol{\theta}}^* \leftarrow \frac{1}{s} \sum_{j=1}^{s} \hat{\theta}(\mathbb{Q}^{(1)*}_{n,j}, \ldots, \mathbb{Q}^{(m)*}_{n,j})$
$\hat{\xi}(Q_n(P)) \leftarrow \frac{1}{s} \sum_{j=1}^{s} \xi(\mathbb{Q}^{(1)*}_{n,j}, \ldots, \mathbb{Q}^{(m)*}_{n,j})$

Similar to the original BLB model, we take the average over all of the subsamples' predicted values to determine the final predicted value for a given test observation. Each subsample's predicted values come from the means of their bootstrap distributions of random forest predictions where each random forest was grown on a different BLB resample.

4.3 Predictions and Confidence Intervals

Now that we know how to construct random forests in the BLB structure, we can move onto what to extract from the model. Instead of using BLB to estimate the mean [Kleiner et al., 2014], here we want to estimate the predictions of test observations using a random forest. The mean and random forest seek to answer different questions about a dataset. A mean simply takes the average response value of a set of data in order to estimate a single parameter from a population, whereas a random forest uses training data to construct a set of rules for predicting the value of an unknown, new observation. That is to say, there is no true population value of a random forest in the same way that there exists a true average value μ. A random forest seeks to find the best prediction of a new observation.

4.3.1 Prediction

It suffices to say that integrating random forests into the BLB structure will involve an additional component that did not exist in the original structure: a new test observation(s) which we will denote as $Z_1, \ldots, Z_m$. Then, when we construct each random forest from each resample, we need to have the new test observation available from which each random forest can predict its value. Rather than have one prediction for a parameter, we produce a vector of predicted values $\bar{\boldsymbol{\theta}} = \bar{\theta}_1, \ldots, \bar{\theta}_m$ for test observations $Z_1, \ldots, Z_m$.

4.3.2 Standard Error

Central to our notion of standard error in the Bag of Little Bootstraps is that we take independent subsamples of our data before bootstrapping. Then, each bootstrapping procedure is independent of all the others because the subsamples are independent from each other. Recall that in the Bag of Little Bootstraps, a random forest is built from a BLB resample of a subsample. This means that a random forest coming from one subsample is entirely independent of another random forest built from a different resample. Because each random forest model is used to predict the response of an incoming test observation, it follows that the averaged predictions from one subsample are also independent from the averaged predictions from another subsample. If our predictions are independent from each other, then we can use them to calculate the appropriate standard error.

Recall that s is the number of subsamples in BLB. Suppose we only have one test observation that we are trying to predict. Instead of using bold vector notation for the predictions, $\hat{\boldsymbol{\theta}}$, we will use $\hat{\theta}$ for this section. Suppose subset j has empirical distribution $\mathbb{Q}^*_{n,j}$ of the bootstrapped random forest predictions. Then $\hat{\theta}^*_j = \hat{\theta}(\mathbb{Q}^*_{n,j})$ is the corresponding prediction for subset j. Then, our final prediction is $\bar{\theta}^* = \frac{1}{s}\sum_{j=1}^{s} \hat{\theta}^*_j$.

After computing predictions from each subsample, we end up with s predictions from one training dataset, which is similar to drawing observations, or making predictions from one random sample. We can then extend the standard notion of standard deviation for a sample and define

$$SD = \sqrt{\frac{1}{s-1}\sum_{j=1}^{s}(\hat{\theta}^*_j - \bar{\theta}^*)^2}. \tag{4.5}$$

Then, the standard error of the predicted value is the following:

$$SE(\bar{\theta}^*) = \frac{SD}{\sqrt{s}}. \tag{4.6}$$

We will use this value of standard error to construct confidence intervals for all predictions coming out of the "BLB-RF" as:

$$\bar{\theta}^* \pm t \cdot \frac{SD}{\sqrt{s}}. \tag{4.7}$$

Chapter 5

Simulations and Results

The main purpose of running simulations is to compare the accuracy of prediction from running traditional random forests on large datasets versus running random forests in the Bag of Little Bootstraps model. I used simulated data to allow for knowledge of the true underlying distribution, P, sampling distributions $Q_n(P)$, and corresponding estimator quality assessments $\xi(Q_n(P))$. Because our focus is on random forests, a sampling distribution $Q_n(P)$ will be a distribution of random forest predictions of some test observation. For simplicity, we will denote a sampling distribution of some ith test observation as Q_i.

5.1 Data and Notation

Our data has the form $X_i = (\boldsymbol{X_i}, Y_i) \sim P$, i.i.d. for $i = 1, \ldots, n$ where $\boldsymbol{X_i} \in \mathbb{R}^d$. Each dataset is generated from a true underlying distribution P that consists of a linear model $Y_i = \boldsymbol{X_i}^T \boldsymbol{\beta}_d + \epsilon_i$, with $d = 10$. The $\boldsymbol{X_i}$ are drawn independently from $\boldsymbol{X_i} \sim \text{Unif}(-5, 5)$ with $\epsilon_i \sim \text{Normal}(0, 1)$. The β coefficients are each drawn independently from $\beta \sim \text{Unif}(-10, 10)$. Therefore, when we generate a prediction of a test observation, we are predicting $E(Y_i) = \boldsymbol{X_i}^T \boldsymbol{\beta}_d$.

Because we are dealing with random forests, our experiment consists of two types of datasets: test and training. For this section, we will only have one test set of size $n_{test} = 500$, and various training sets of size $n_{train} = 1000$.

The two models we are comparing are the BLB-RF model, as described in Chapter 4, and bootstrapping traditional random forests, as described in Chapter 3. The training sets are used to construct either model.

We can then use our models to generate predictions of our test data. Our test data $Z_1, \ldots, Z_{500}$ comes from our previously defined underlying distribution P. Our estimator $\bar{\boldsymbol{\theta}} = \bar{\theta}_1, \ldots, \bar{\theta}_{500}$ is a vector of predictions for our test data where each prediction $\bar{\theta}$ is generated from either a random forest γ or BLB-RF. We will define ξ as a procedure that computes a 95% confidence interval of a prediction from a certain model. In particular, given a sampling distribution of random forest predictions $Q_n(P)$, ξ computes the standard deviation σ of $Q_n(P)$.

5.1.1 BLB-RF Model

Recall that when we create a BLB-RF model from a training dataset, we need both the training and test set because our output will consist of prediction estimates of the test data, and corresponding estimates of standard error. We describe how to generate prediction and standard error estimates in Chapter 4.

5.1.2 Bootstrapping Random Forests model

The traditional random forest model also requires both a training set from which to construct a model, and test data whose response values we want to predict. Because BLB is a type of modified bootstrapping procedure, we will compare it to bootstrapping where our statistic of interest is the prediction from a random forest. We describe how to compute predictions in Chapter 3.

5.2 Procedure and Plots

In order to evaluate the estimates of ξ and $\bar{\boldsymbol{\theta}}$ from a random forest model to that of our BLB-RF model, we will compare the BLB-RF estimates and estimates from traditional bootstrapping to an approximation of the true values of $\xi(Q_n(P))$, where $Q_n(P)$ is the a sampling distribution of random

forest predictions for some test observation.

First, we construct an approximation to the true sampling distribution of random forest predictions of our test data. Then, we generate random forest predictions and corresponding standard error estimates from traditional bootstrapping, and compare them to the approximation of the true sampling distributions. Finally, we do the same for a BLB-RF model.

5.2.1 Generating an approximation to $\xi(Q_n(P))$

We can approximate a sampling distribution for each of the 500 test data points by generating 2,000 realizations of datasets of size $n_{train} = 1000$ from P.

On each of the 2,000 training samples of size $n_{train} = 700$ we do the following:

1. Build a random forest γ.
2. For each test observation Z_i for $i = 1, \ldots, 500$:
    - Generate prediction $\hat{\theta}_i = \gamma(Z_i)$.

Then, each test observation has a corresponding approximation to sampling distribution Q_i such that we end up with approximations of 500 sampling distributions.

Each ith test observation Z_i has 2,000 predictions $\hat{\theta}_{i,1}, \ldots, \hat{\theta}_{i,2000}$. The center of sampling distribution Q_i of test observation Z_i is the average over all 2,000 predictions, $\bar{\theta}_i = \frac{1}{2000} \sum_{j=1}^{2000} \hat{\theta}_{i,j}$. Recall that the true standard error is just the standard deviation of the sampling distribution. Our confidence interval $\xi(Q_i)$ for some test observation Z_i is $\hat{\theta}_i \pm z_\alpha \sigma$. For each test observation, we would hope that $Y_i = \boldsymbol{X}_{\boldsymbol{i}}^T \boldsymbol{\beta}_d$, would be contained in the confidence interval.

The plot below shows the confidence intervals and predictions of each test observation's sampling distribution. Each point is a prediction of a test observation. The x-axis is our truth, and the y-axis is our predictions such that we want to capture the dotted line $y = x$. The blue confidence intervals are those confidence intervals that actually capture the truth. Only 242 out of the 500 test observations captured the truth. The "regression to the

mean" effect is very noticeable in this plot, as the blue confidence intervals are centered around 0.

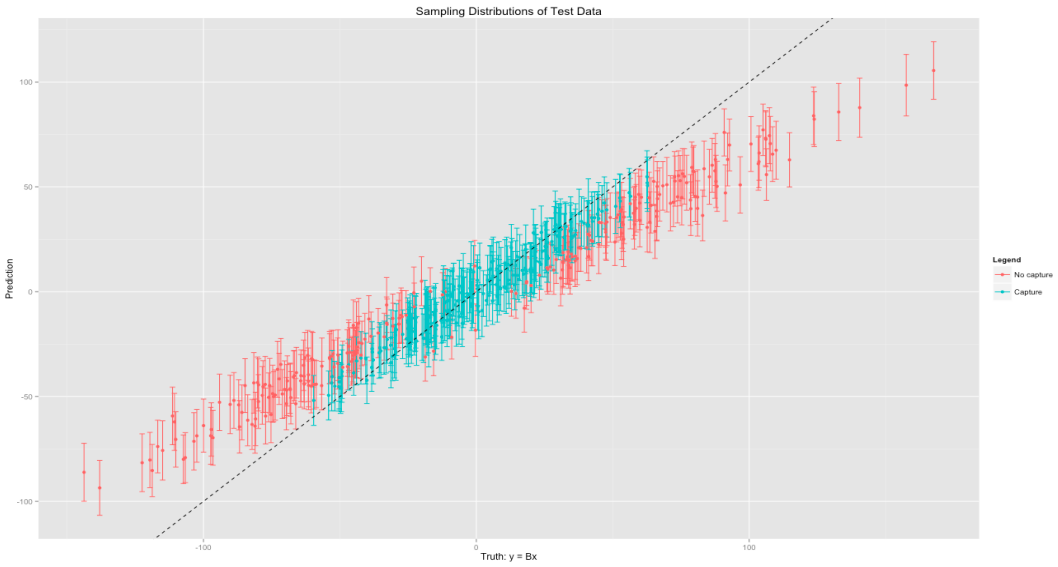

Figure 5.1: Confidence intervals of sampling distributions of random forest test predictions.

5.2.2 Generating predictions from a bootstrapping model

The purpose of bootstrapping is to give us information about a theoretical sampling distribution of some statistic. For the case of random forests, we can bootstrap from one training sample and generate a random forest prediction on each bootstrap resample to create a bootstrap distribution which we can compare our BLB-RF model to. Because BLB is supposed to maintain the same statistical properties as bootstrapping, we would hope that both bootstrapping random forests and BLB-RF produce similar information about the sampling distributions which we approximated in the previous section.

To bootstrap, we used one training sample of size $n_{train} = 1000$ to draw 200 bootstrap resamples. On each of the 200 bootstrap resamples, we do the following:

1. Build a random forest γ.

2. For each test observation Z_i for $i = 1, \ldots, 500$:

    - Generate prediction $\hat{\theta}_i = \gamma(Z_i)$.

Then, each test observation has a corresponding bootstrap distribution.

Each ith test observation Z_i has 200 predictions $\hat{\theta}_{i,1}, \ldots, \hat{\theta}_{i,200}$. The bootstrap distribution of test observation Z_i is the average over all 200 predictions, $\bar{\theta}_i = \frac{1}{200} \sum_{j=1}^{200} \hat{\theta}_{i,j}$. Our standard error estimate is just the standard deviation of the bootstrap distribution. Our confidence interval for some test observation Z_i is $\hat{\theta}_i \pm z_\alpha \sigma$. For each test observation, we would hope that $Y_i = \boldsymbol{X_i}^T \boldsymbol{\beta}_d$, would be contained in the confidence interval.

The plot below shows the confidence intervals and predictions of each test observation's bootstrap distribution. Each point is a prediction of a test observation. The x-axis is our truth, and the y-axis is our predictions such that we want to capture the dotted line $y = x$. The blue confidence intervals are those confidence intervals that actually capture the truth. Only 197 out of the 500 test observations captured the truth. The "regression to the mean" effect is very noticeable again, according to the same trend as in Figure 5.1. Because the regression to the mean is present in every plot, we will assume that this is not due to the BLB-RF model, but either the population P from which we are drawing our data, or random forest predictions.

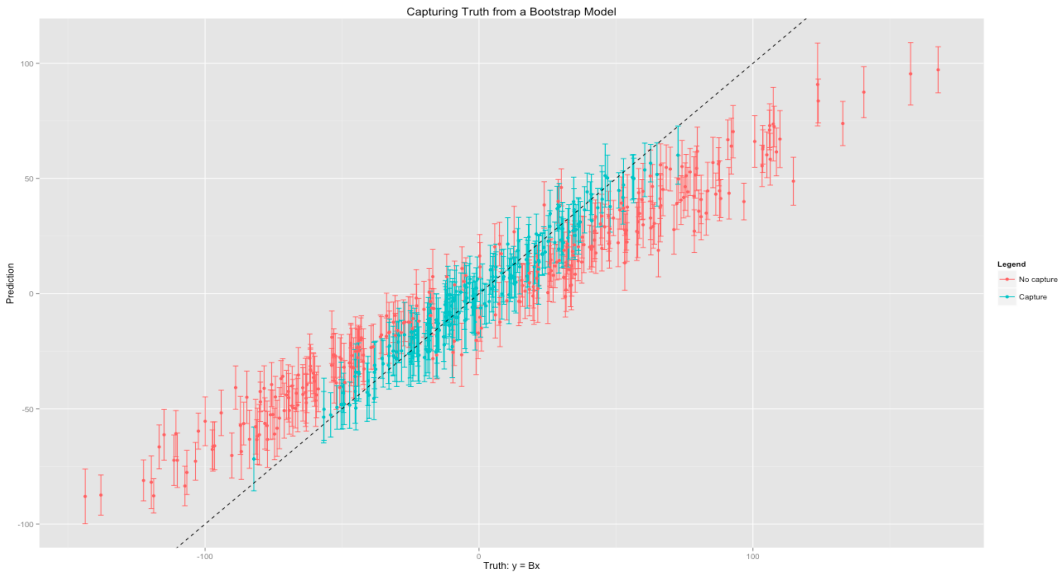

Figure 5.2: Confidence intervals of bootstrap distributions of random forest test predictions that capture truth.

A bootstrap distribution is not necessarily centered around the center of its corresponding theoretical sampling distribution, but a bootstrap confidence interval is theoretically supposed to capture the center of its corresponding sampling distribution. Therefore, we expect more of the confidence intervals from bootstrap distributions to capture the sampling distribution centers than the truth. The plot below is similar to that in Figure 5.2, except that the blue confidence intervals are those that capture the sampling distribution means rather than the truth. As expected, more confidence intervals capture the centers of their corresponding sampling distributions than those that capture truth. In Figure 5.3, 405 out of the 500 confidence intervals capture their corresponding sampling distribution's mean. This is almost double the capture rate of capturing the truth, as expected.

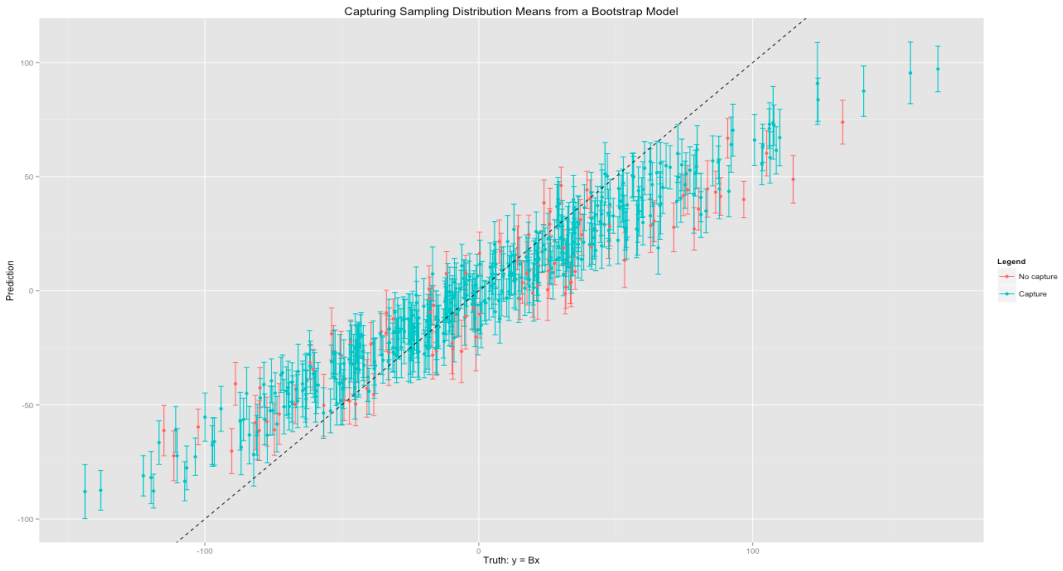

Figure 5.3: Confidence intervals of bootstrap distributions of random forest test predictions that capture sampling distribution means.

5.2.3 Generating predictions from a BLB-RF model

Because BLB is supposed to be a means of assessing the quality of estimators similar to bootstrapping, we can also compare our estimates of standard error from BLB, $\hat{\xi}$, with our approximations to the true $\xi(Q_i)$ for $i = 1, \ldots, 500$. We constructed one BLB-RF model from the same training dataset used for the previous section (size $n_{train} = 1000$) and generated predictions and estimates of standard error (as in Equation 4.6) to create a confidence interval (as in Equation 4.7) for each test observation.

The plot below shows the BLB-RF confidence intervals and predictions of each test observation. Each point is a BLB-RF prediction of a test observation. The x-axis is our truth, and the y-axis is our predictions such that we want to capture the dotted line $y = x$. The blue confidence intervals are those confidence intervals that actually capture the truth. Only 99 out of the 500 test observations captured the truth. This is much fewer than the capture rate of the sampling distributions, and almost half the number of confidence intervals that capture truth from bootstrapping. Similarly, the

"regression to the mean" effect is also very noticeable in this plot, as the blue confidence intervals are centered around 0.

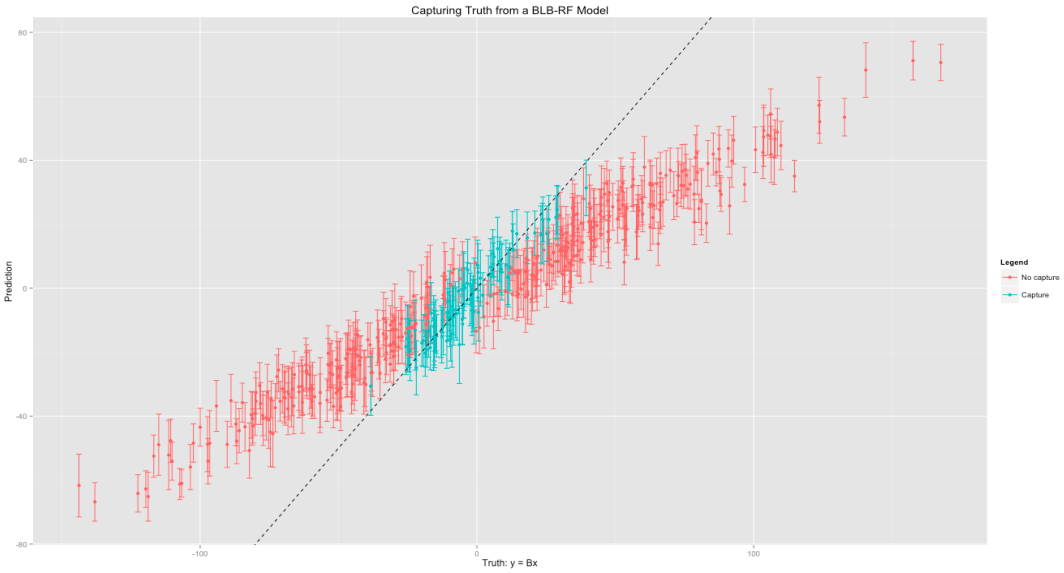

Figure 5.4: Confidence intervals of BLB-RF test predictions that capture truth.

Similarly with bootstrapping, we can assess how well confidence intervals capture the centers of their corresponding sampling distributions. The plot below is similar to that in Figure 5.4, except that the blue confidence intervals are those that capture the sampling distribution means rather than the truth. As expected, more confidence intervals capture the centers of sampling distributions than those that capture truth. In Figure 5.5, 242 out of the 500 confidence intervals capture their corresponding sampling distribution's mean.

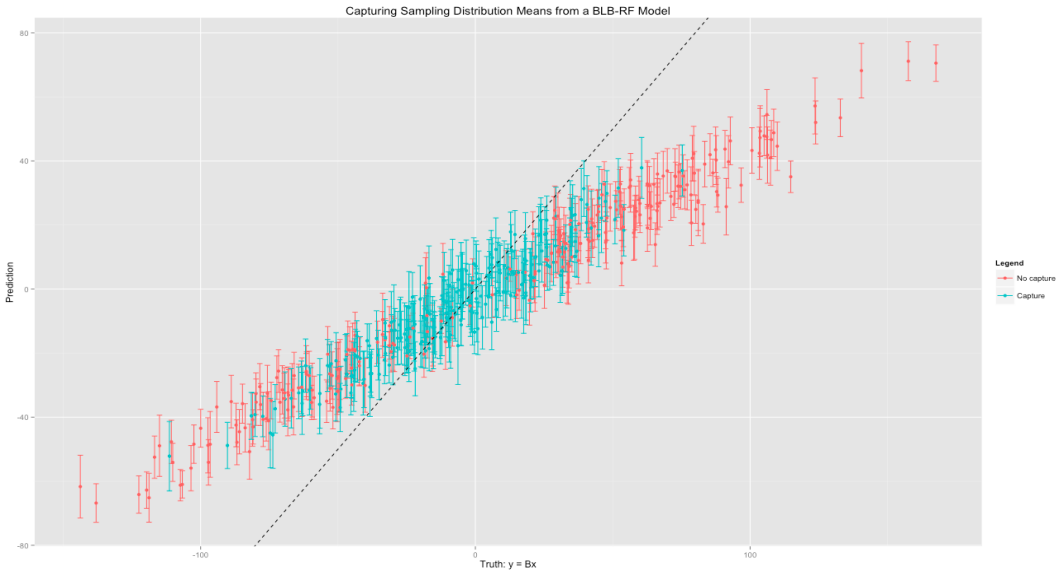

Figure 5.5: Confidence intervals of BLB-RF test predictions that capture truth.

Below is a summary of confidence interval capture rates from the bootstrapping model and the BLB-RF model, both of which were constructed from the same training dataset of size $n_{train} = 1000$. It is obvious that bootstrapping does a lot better, but it is important to note that both follow the same trend of capturing more sampling distribution means than truth values.

Capture	Bootstrap	BLB-RF
Truth	197	99
Samp Dist	405	242

Table 5.1: Table to compare confidence interval capture rates.

5.2.4 Comparing standard error estimates

To see how the standard error estimates compare to our approximations of true standard error from the sampling distributions, we can assess the deviation of BLB-RF standard error estimates (according to section 4.3.2) and bootstrapping standard error estimates from the true standard errors $\sigma(Q_i)$.

Let c be the estimated standard error from either BLB-RF or bootstrapping and let c_o be the true standard error from a sampling distribution. Then we measure the deviation of c from c_o as $|c - c_o|/c_o$.

The plot below shows the distribution of deviance values. The red are the deviance values from the bootstrapping standard error estimates, and the blue are the deviance values from the BLB-RF standard error estimates. We can see that the deviance values are right skewed for both BLB-RF and bootstrapping, and range from 0 to around 0.8, which means that the standard error estimates are very comparable to the approximations to the true standard error values of our sampling distributions. Bootstrapping does slightly better, as expected.

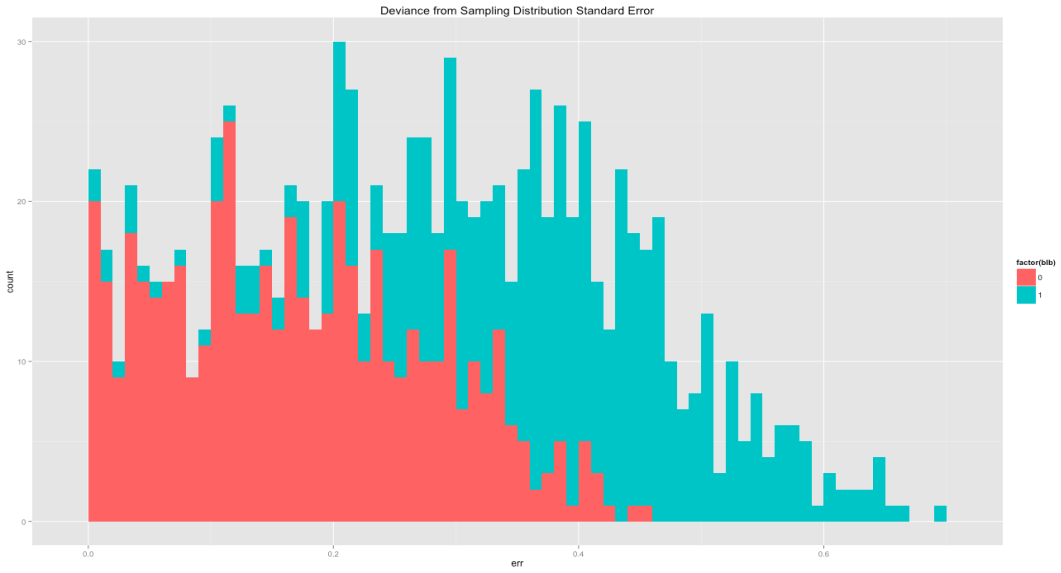

Figure 5.6: Comparing Deviance from Sampling Distribution Standard Error between BLB-RF and bootstrapping estimates.

5.2.5 Discussion

Our goal was to assess how BLB-RF standard error estimates compare to the true standard error values from sampling distributions of random forest predictions, and to estimates and predictions from traditional bootstrapping.

Recall that bootstrapping is a resampling tool that is used to approximate the shape and spread of a theoretical sampling distribution. Because BLB is purported to have the same statistical properties as bootstrapping, our BLB-RF model should also approximate the shape and spread of a theoretical sampling distribution just as a bootstrap distribution does. From looking at the deviance values of our BLB-RF standard error estimates, we can see that our BLB-RF model has similar behavior to that of traditional bootstrapping, but just not as good. Because I only compared BLB-RF with bootstrapping on one training set, this simulation isn't entirely exhaustive, and likewise, n_{train} is relatively small, and there are obvious regression to the mean effects in all plots that are probably interfering with prediction accuracy. However, it is promising that our BLB-RF model behaves somewhat similarly to bootstrapping, but on a smaller scale. Ideally, we would be able to run this simulation on a much larger scale, because BLB is to be utilized for processing high-dimensional, massive datasets.

Chapter 6

Conclusion

The Bag of Little Bootstraps is a promising and new resampling model to accommodate large datasets. From a preliminary simulation exercise, we can see that integrating the random forest prediction model for regression into the BLB structure (BLB-RF) approximately maintains the bootstrapping property of estimating standard error from a sampling distribution. Because the BLB-RF model was not tested on a parallel computing architecture, there were limitations in computing time (a rather counter-intuitive and ironic barrier) such that I could not test the BLB-RF model on large datasets, which is a situation that BLB is meant for. In the future, I would like to build a computing structure that does run BLB-RF in parallel such that it can be tested on large datasets, and be accompanied with a more thorough analysis of its accuracy. Large datasets are being utilized everywhere for all types of operations so finding solutions to process and compute on large datasets more quickly and efficiently is an extremely relevant problem. The

6.1 Acknowledgements

Thank you to the Professor Jo Hardin and the Pomona College Mathematics Department for giving me the opportunity to write this. I owe my life to my family, my friends, Dialynas 350, my poteto, the Asian American Resource Center, and the Asian American Mentor Program.

Bibliography

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

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

[James et al., 2014] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2014). *An Introduction to Statistical Learning*, volume 74 of *Springer Texts in Statistics*. Springer.

[Kleiner et al., 2014] Kleiner, A., Talwalkar, A., Sarkar, P., and Jordan, M. I. (2014). A scalable bootstrap for massive data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76:795–816.

[P.J. Bickel and van Zwet, 1997] P.J. Bickel, F. G. and van Zwet, W. (1997). Resampling fewer than n observations: Gains, losses, and remedies for losses. *Statistica Sinica*.

[Politis et al., 1999] Politis, D. N., Romano, J. P., and Wolf, M. (1999). *Subsampling*. Springer.