2013

Improvements to random forest methodology

Ruo Xu
Iowa State University

Improvements to random forest methodology

by

Ruo Xu

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Statistics

Program of Study Committee:

Dan Nettleton

Daniel J. Nordman, Major Professor

Stephen B. Vardeman

Kenneth Koehler

Huaiqing Wu

Iowa State University

Ames, Iowa

2013

TABLE OF CONTENTS

ABSTRACT

Random forest (RF) is a widely used machine learning method that shows competitive prediction performance in various fields, including biological science, finance, chemical engineering, agroscience, medical analysis, etc. In this dissertation, we study some characteristics and modifications of RFs in order to improve its prediction performance.

In CHAPTER 1, we review the mechanics of classification and regression trees (CARTs), bootstrap aggregation (bagging) and RFs. The properties of RFs are discussed, along with several variations of this method.

In CHAPTER 2, we describe a counter-intuitive discovery using RFs: the out-of-sample prediction errors can be reduced by augmenting the regressor with a new scientifically meaningless predictor variable independent of all variables in the dataset. We explain this phenomenon using a simulated example and discuss the importance of this result in interpreting predictor variable importance in RFs.

RF predictions can be biased. In CHAPTER 3, we apply an iterative debiasing approach based on bagging to RFs and test this bias correction method with real datasets. The debiasing approach can significantly improve RF predictions. The number of debiasing iterations can be tuned using cross-validation.

Standard RF methodology generates a common RF from a given training sample, regardless of test cases. In CHAPTER 4, we propose a new way to grow a RF specifically predicting a particular test case, namely, Case-Specific Random Forests (CSRF). We also suggest Case-Specific Variable Importance (CSVI), a new definition of predictor variable importance in terms of the prediction performance on a particular test case.

Prediction error estimation is generally useful in evaluation of a prediction rule. All present methods deal with estimating prediction errors averaging over the distribution of a test set. In CHAPTER 5, we propose a method to estimate expected prediction loss on a specific regressor point using RF methodology.

CHAPTER 1. GENERAL INTRODUCTION

1 Introduction

Random forest (RF) methodology is a machine learning technique useful for prediction problems. The RF algorithm, developed by Leo Breiman (2001a), applies bootstrap aggregation (bagging) (Breiman 1996a,b) and random feature selection (Ho 1995, 1998; Amit and Geman 1997) to individual classification or regression trees for prediction. There are many studies showing that RFs have impressive predictive performance in regression and classification problems in various fields, including financial forecasting, remote sensing, and genetic and biomedical analysis (Siroky 2009; Kumar and Thenmozhi 2006; Shi et al. 2005; Ward et al. 2006; Diaz-Uriarte and de AndršŠs 2006; Jiang et al. 2007; Pal 2003; Palmer et al. 2007; Goldstein et al. 2011). The RF method has been compared with other learning methods, such as partial least squares regression, support vector machine and neural networks in Breiman (2001a), Kumar and Thenmozhi (2006), Diaz-Uriate and Alvarez de Andrez (2006), and Palmer et al. (2007). In such comparisons, RFs typically shows comparable or even better prediction performance. Besides the appealing prediction performance by RFs, the availability of computation package *randomForest* in R (Liaw and Wiener 2002) is another reason for its great popularity. To date, Breiman's original paper (2001a) has been cited over 3700 times according to Web of Science. A good resource for using RFs is Leo Breiman's website http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm.

2 Section Organization

In this thesis, we study properties of random forests and the approaches to improve the prediction performance by RFs as outlined in the thesis abstract. In order to present modifications for improving RFs and understand the exposition to follow, it is first helpful to provide an overview of the basic mechanics and properties of RFs, which we seek to do in this chapter.

A RF is an ensemble of tree predictors, with each independently and randomly generated tree depending on the values of a random vector (Breiman 2001a). To appreciate the RF algorithm, it is initially helpful to understand the mechanism of growing a single decision tree. In Section 3, we will introduce how a single tree is generated. In Section 4, we describe the algorithm of growing a RF, which includes a series of randomly generated trees. In Section 5, we discuss various byproducts of RFs, such as the proximity measure and measure of predictor variable importance, which are important in later chapters for developing modifications and improvements to RFs. We also discuss how a RF deals with predictor

variables with missing values. In Section 6, we introduce some useful extensions of the RF method.

3 Introduction to Classification and Regression Tree

Classification and Regression Trees (CARTs) belong to the family of binary tree structured regressors or classifiers (Breiman et al. 1984). Understanding how a CART is generated is critical in order to understand how a RF works. Thus, we first review the mechanism of CARTs. Most of the material is adopted from (Breiman et al. 1984). The algorithm-based approach for creating a CART involves repeated binary splits of sets of objects, namely nodes or leaves, into two descendant subsets (called subnodes or subleaves). For each split, one predictor variable and a corresponding "cut-point" (chosen according to some within-subnode homogeneity criterion described in Subsection 3.2) are used to partition a node into two subnodes.

3.1 Ways to Deal with Different Predictor Types

If a predictor variable x_j is numerical, the candidate cut-points are the unique midpoints of the intervals between ordered values from this predictor (from the available cases in a given node). Then node cases are partitioned into two subnodes depending on whether the value of x_j is below or above the cut-point. For example, suppose in a node S containing five cases, the values of x_j are -1.0, 1.0, 1.0, 2.8 and 3.6. Then the corresponding unique midpoints are 0, 1.9 and 3.2. So three possible binary splits by x_j are "if $x_j > 0$ or not", "if $x_j > 1.9$ or not" and "if $x_j > 3.2$ or not". We see it is the order of x_j values but not their actual values that matters.

If a predictor variable x_j is categorical without ordinality, dummy variables need to be made. For an x_j with G categorical levels in node S, there will be $2^{G-1} - 1$ possible binary splits. For example, if the cases in a node S have the three unique categories, *red*, *blue* and *green*, then we can split node S on x_j in three ways, "if *red* or not", "if *blue* or not" and "if *green* or not".

3.2 Grow a Tree Iteratively by Measuring Node Heterogeneity

The fundamental idea of node splitting is to make each subnode as homogeneous as possible after the current splitting. Hence CART is a greedy method. The function for heterogeneity measure must be a concave function, so that the function quantity will never increase after any split (Coppersmith et al. 1999). Denote the parent node to be split as S, and the left and right subnodes as L and R. For regression trees, the sum of squared errors (SSE) is typically used to measure the heterogeneity of a node S,

$$I_S = \sum_{i \in S} (y_i - \bar{y})^2, \tag{1}$$

where $\bar{y}$ is the average of response values in node S. Then the decrease of heterogeneity of node S after splitting into L and R is

$$\Delta I(S, L, R) = I_S - (I_L + I_R). \tag{2}$$

For classification trees, the Gini index of diversity or entropy is often used to reflect the impurity of a node S. Suppose there are G categories in the data. With $\pi_S(g)$ defined as the proportion of the observations from the gth category in node S, and n_S as the number of cases in node S, the Gini index is defined as

$$n_S \cdot \sum_{g=1}^{G} \pi_S(g)\,[1 - \pi_S(g)], \tag{3}$$

and the entropy is defined as

$$n_S \cdot \sum_{g=1}^{G} \pi_S(g)\,[-\log(\pi_S(g))]. \tag{4}$$

Taking I_S to be either (3) or (4), the decrease of impurity of node S after split is defined by (2) as in the regression case. In the standard CART approach, the chosen split for each node is the one that maximizes $\Delta I(S, L, R)$ in (2). This process is repeated until a largest possible tree is obtained and no more nodes can be split.

3.3 Prediction by a Tree

Once a tree is grown, we need to assign a predicted value for each terminal node. Suppose a terminal node $Node_k$ contains n_k training cases $C_1, \ldots, C_{n_k}$ with $C_i \equiv (\boldsymbol{x}_i, y_i)$, then the prediction for a given terminal node is a weighted average of all the response values in this node, $y_1, \ldots, y_{n_k}$,

$$\hat{y}_k = \frac{1}{\sum_{i=1}^{n_k} w_i} \sum_{i=1}^{n_k} w_i \cdot y_i, \tag{5}$$

where w_i, $i = 1, \ldots, n_k$ are nonnegative numbers. In regression problems,

$$w_i = \frac{1}{n_k}.$$

In classification problems,

$$w_i = \begin{cases} 1 & \text{if } \sum_{l=1}^{n_k} \mathcal{I}(y_l = y_i) \geq \sum_{l=1}^{n_k} \mathcal{I}(y_l = y_j) \, \forall y_i \neq y_j, \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathcal{I}(\cdot)$ denotes the indicator function. This means the prediction for a regression problem is simply the arithmetic average of all training case responses in $Node_k$, while that for classification is the category receiving a plurality of votes by all training cases in the node.

Because a CART is a way to partition the regressor space to disjoint hyperrectangles, a test case with given predictor values will end up in one terminal node. All test cases falling in a terminal node $Node_k$ will be predicted using $\hat{y}_k$ in (5).

3.4 Pruning a Tree

Usually the largest tree T_{max}, in which there are no more splits that can be done, performs badly for prediction of a new test dataset, because the variance of prediction increases as the size of the tree exceeds the optimal. Accordingly, pruning is conducted to avoid this overfitting problem. For a tree T with K terminal nodes, let $R(T)$ denote the overall training sample error of this tree. Let α be a complexity parameter, which quantifies the penalty for increasing the size of the tree by splitting one node. Define the cost-complexity measure of tree T as

$$R_\alpha(T) = R(T) + \alpha K. \tag{6}$$

$R_\alpha(T)$ is formed by adding a penalty on tree complexity to the training error $R(T)$. For a given α,

define $T(\alpha)$ to be the tree that minimizes $R_\alpha(T)$ over all trees T that are subtrees of T_{max} (i.e., trees that can be further split to obtain T_{max}). $T(\alpha)$ has the following properties: i) $T(\alpha)$ is a subtree of T_{max}; ii) $T(\alpha)$ yields the least cost-complexity measure R_α among all the subtrees of T_{max}; iii) $T(\alpha)$ yields the least training error R among all the subtrees of T_{max} containing same number of terminal nodes $T(\alpha)$.

As α increases, the minimizing tree $T(\alpha)$ has fewer terminal nodes. The largest tree T_{max} has only a finite number of subtrees. Hence if $T(\alpha)$ is the minimizing tree for some α, it continues to be the minimizing tree as α increases until a jump point α' is reached, although the complexity parameter α is a positive continuous number. For example, it could be that some tree T_a is the minimizing tree for $\alpha \in (0.1, 0.2]$, and another smaller tree T_b is minimizing for $\alpha \in (0.2, 0.25]$. It can be proved that the sequence of minimizing trees $\{T_j(\alpha_j),\ j = 1, 2, \ldots\}$ is a nested tree sequence, which means that $T_p(\alpha_p)$ is a subtree of $T_q(\alpha_q)$ for $\alpha_p > \alpha_q$. This nested sequence characteristic makes the pruning process computation feasible. Instead of checking all the subtrees of T_{max} which can be an extremely large number, we only need to look at this nested sequence of minimizing trees.

What value should α take? The goal of pruning is to select a tree from $\{T_j(\alpha_j),\ j = 1, 2, \ldots\}$ that gives the lowest *out-of-sample* prediction error for an independent dataset. Cross-validation is a preferred method for this purpose. For a V-fold cross-validation, the training sample is randomly separated into V subsets, each containing the same or nearly the same number of elements. Let $subset_v$ denote the vth of the V subsets. Let $sample_v$ denote the training sample obtained by combining all subsets except $subset_v$.

In a V-fold cross-validation, the largest possible tree is grown for each of the V training samples, $sample_1, \ldots, sample_V$. Let $T_{max,\,v}$ denote the largest tree grown for $sample_v$, $v = 1, \ldots, V$. From $T_{max,\,v}$, the nested minimizing tree sequence $\{T_j(\alpha_j),\ j = 1, 2, \ldots\}$ can be obtained, and $subset_v$ serves as an independent test sample for each minimizing tree $T_{j,\,v}(\alpha_j)$. An prediction error $R_{CV}(\alpha)$ for all V minimizing trees can be calculated as a function of α. The optimal complexity parameter α_{CV} corresponds to the size of the tree producing the lowest $R_{CV}(\alpha)$.

Once a CART is pruned, a prediction value is assigned to each terminal node as explained in Subsection 3.3. An example of classification tree with two categories is adopted from Hammann et al. 2010 and is shown in Figure 1.

Figure 1: An example of a classification tree.

4 Introduction to Random Forests and Extensions

4.1 Bootstrap Aggregation (Bagging)

A large decision tree may lead to large prediction variance. Pruning alleviates overfitting, but it was Breiman's bootstrap aggregation method (bagging, Breiman 1996a; Buhlmann and Yu 2002; Friedman and Hall 2000) that effectively solved the overfitting problem. Bagging is an ensemble learning method. Other ensemble methods include boosting, stacking, and bumping (Freund and Schapire 1996; Hastie et al. 2009). Bagging has been shown able to effectively reduce prediction variance for *unstable* prediction rules. The term *instability* was explained heuristically by (Breiman 1996b) and then more systematically defined by (Buhlmann and Yu 2002), according to whom, a statistic $\hat{\theta}_n$ is called stable if

$$\hat{\theta}_n \quad \rightarrow \quad \theta \text{ as } n \rightarrow \infty,$$

for some fixed value θ, where θ is not necessarily $E(\hat{\theta}_n)$. Buhlmann and Yu (2002) argued that unstable prediction rules arise mainly when hard-thresholding decisions with indicators are involved. A decision tree is nothing but partitioning the regressor space with disjoint hyper-rectangles. The prediction by a decision tree is a weighted average of all responses from a terminal node (Subsection 3.3). Therefore the predicted value for some new case can be written in the form of a step function,

$$\hat{y}_i \quad = \quad \hat{y}_k, \text{ if } i \in Node_k.$$

By Buhlmann and Yu's definition (2002), a CART is not a stable prediction rule. By averaging results from bootstrap resamples, bagging smoothes the response surface and hence reduce the prediction variance.

Suppose we have a training set $\boldsymbol{\mathcal{C}} = \{C_1, \ldots, C_n\}$, then the bagging algorithm is as follows,

1) Take M bootstrap resamples of $\boldsymbol{\mathcal{C}}$, say $\boldsymbol{\mathcal{B}}_1, \ldots, \boldsymbol{\mathcal{B}}_M$.

2) For each resample $\boldsymbol{\mathcal{B}}_m$, $m = 1, \ldots M$, fit the model as done for the original training data $\boldsymbol{\mathcal{C}}$ to obtain $\hat{f}^*_m$, the fitted prediction rule that maps the predictor space to the response space.

3) Then the prediction for a new case C_0 with covariate $\boldsymbol{x}_0$ is

$$\tfrac{1}{M}\sum_{m=1}^{M} \hat{f}^*_m(\boldsymbol{x}_0) \quad \text{for regression problems}$$

and

$$argmax_g\{\textstyle\sum_{m=1}^{M} \mathcal{I}\,[\hat{f}^*_m(\boldsymbol{x}_0) = g]\,\} \text{ for classification problems.}$$

4.2 Random Subspace Method

Two factors affect the prediction errors of an ensemble of trees: (1) how accurate the individual trees are, and (2) how dissimilar the trees are from each other. Factor 1 is responsible for prediction bias by the whole forest of trees, while factor 2 is related to prediction variance of the forest. The bagging approach avoids generating exactly the same trees by resampling with replacement, which in turn reduces prediction variance and alleviates overfitting by the tree ensemble. Apart from this, there are other ways to create dissimilar trees.

Ho (1995; 1998) and Amit and Geman (1997) independently proposed to grow trees only on a randomly chosen subspace, instead of the whole regressor space. The authors showed that the combined tree classifier grown using random subspaces was better than individual trees in terms of prediction errors. There are many ways to apply the random subspace method in decision trees, including randomly choosing a subset of predictors at the tree level or at the split level, randomly choosing a subset of cut-points for a predictor variable before splitting, etc. Geurts et al. (2006) proposed extremely randomized trees, where splits (both predictor variable and its cut-points) are selected completely at random. Many machine learning methods balance the "bias-variance trade-off," so that a satisfactory prediction error can be obtained. Similarly RFs uses the bagging approach and the random subspace idea to substantially smooth prediction and reduce prediction variance, without sacrificing prediction accuracy too much. When a split is to be made, only

a subset of predictor variables are randomly selected and considered as split candidates. Our experience shows that a random selection of predictors at the split level is better than a selection at the tree level.

4.3 The Random Forest Algorithm

Now we are ready to describe the RF algorithm. Suppose we have a training set $\boldsymbol{\mathcal{C}} = \{C_1, \ldots, C_n\}$ with $C_i \equiv (\boldsymbol{x}_i, y_i)$ and an independent test case C_0 with predictor $\boldsymbol{x}_0$.

1) Sample the training set $\boldsymbol{\mathcal{C}}$ with replacement to generate bootstrap resamples $\boldsymbol{\mathcal{B}}_1, \ldots, \boldsymbol{\mathcal{B}}_M$.

2) For each resample $\boldsymbol{\mathcal{B}}_m$, $m = 1, \ldots, M$, grow a classification or regression tree T_m as described in Section 3, except for the following modifications.

a) At each split, only predictors in a randomly selected subset of predictors are considered as discussed in Section 4.2. Let p denote the total number of predictor variables in $\boldsymbol{\mathcal{C}}$. Breiman suggested using $\lfloor p/3 \rfloor$ as this number, which is the default set up in the R package *randomForest*.

b) Each tree is grown until all nodes contain observations no more than the *maximal terminal nodesize*, *MTN*, a pre-specified parameter. Unlike CART, trees in RFs are not pruned.

3) For predicting the test case C_0 with covariate $\boldsymbol{x}_0$, the predicted value by the whole RF is obtained by combining the results given by individual trees. Let $\hat{f}^*_m(\boldsymbol{x}_0)$ denote the prediction of C_0 by mth tree, the RF prediction is

$$\begin{cases} \frac{1}{M}\sum_{m=1}^{M} \hat{f}^*_m(\boldsymbol{x}_0) & \text{for regression problems} \\ argmax_g\{\sum_{m=1}^{M} \mathcal{I}\left[\hat{f}^*_m(\boldsymbol{x}_0) = g\right]\} & \text{for classification problems.} \end{cases} \tag{7}$$

5 Properties of Random Forests

5.1 Tuning Random Forests

Breiman (2001a) first argued that the number of trees in a forest is a complexity parameter and tried to prove that overfitting could be overcome by growing more trees. However, it is the maximal terminal nodesize (MTN) not the number of trees that is a complexity parameter of RFs. A RF is fundamentally an adaptive k-nearest neighbor predictor (Lin and Jeon 2006). For a single decision tree, the MTN value specifies the k value corresponding to the k-nearest neighbor method, which controls the complexity of the decision tree. Growing an ensemble of trees makes the prediction function (7) smoother than that of an individual tree, but the MTN value still controls the depth of each tree, and in turn controls the complexity of the whole forest. In the R package *randomForest*, the default MTN value is 5 for regression and 1 for classification. In data analysis, the optimal MTN value can be found using cross-validation or

bootstrap-based error estimation techniques.

5.2 Out-of-bag Predictions

Cross-validation and bootstrap are two major techniques to non-parametrically estimate prediction error for a given model or statistical method (Borra and Ciaccio 2010; Hastie et al. 2009). When the size of a training sample $\boldsymbol{\mathcal{C}}$ is not small, a bootstrap resample $\boldsymbol{\mathcal{C}}_m$ almost always contains only a subset of the original sample. Therefore the model fitted by $\boldsymbol{\mathcal{C}}_m$, $\hat{f}^*_m$, can be used to predict the cases in $\boldsymbol{\mathcal{C}}\backslash\boldsymbol{\mathcal{C}}_m$, and a prediction error estimate is obtained by averaging the out-of-sample errors on all training cases (Efron 2004, Efron and Tibshirani 1997).

A RF is generated with bootstrap samples of the training set, which naturally enables the implementation of the error estimation based on bootstrap. In the RF literature, the predictions of $\boldsymbol{\mathcal{C}}\backslash\boldsymbol{\mathcal{C}}_m$ by $\hat{f}_m$ are called *out-of-bag (OOB) predictions*, and the prediction error estimate is called *out-of-bag prediction error*. These two quantities are formally defined as follows,

1) Draw B bootstrap resamples of the training set $\boldsymbol{\mathcal{C}}$, namely $\boldsymbol{\mathcal{C}}_m$, $m = 1, \ldots, M$. Grow a tree by $\boldsymbol{\mathcal{C}}_m$, denoted as $\mathcal{T}_m$.

2) For any training case $C_i \in \boldsymbol{\mathcal{C}}$, denote $\boldsymbol{\mathcal{O}}_i$ as the set of indices of trees which contain contain C_i. Let $\boldsymbol{\mathcal{O}}_{-i}$ be $\boldsymbol{\mathcal{C}}\backslash\boldsymbol{\mathcal{O}}_i$.

3) The OOB predicted value for C_i with covariate $\boldsymbol{x}_i$ is

$$\tilde{y}_i^{OOB} \quad = \quad \frac{1}{||\boldsymbol{\mathcal{O}}_{-i}||} \sum_{m \in \boldsymbol{\mathcal{O}}_{-i}} \hat{f}^*_m(\boldsymbol{x}_i),$$

where $||\boldsymbol{\mathcal{O}}_{-i}||$ is the size of set $\boldsymbol{\mathcal{O}}_{-i}$.

4) If we use a loss function $L(\cdot)$, the OOB prediction error is

$$\widetilde{Err}^{OOB} \quad = \quad \frac{1}{n} \sum_{i=1}^{n} L(y_i, \tilde{y}_i^{OOB}).$$

In the R package *randomForest*, the OOB prediction and OOB prediction errors (mean squared error for regression and misclassification error for classification) are provided.

5.3 Proximity Measure

As previously mentioned, RFs belong to k-nearest neighbor predictor (Lin and Jeon 2006). The tree splitting algorithm tends to gather similar cases together in a node because each split always maximizes the decrease of heterogeneity in response values. So it is reasonable to consider cases in the same terminal nodes of a tree to be closer in proximity to each other than cases that lie in different terminal nodes. Then the relative frequency of this coincidence over all trees provides a measure of proximity between any pair of cases. The proximity measure in RF literature is defined as follows,

1) A RF of M trees is generated on a training set $\boldsymbol{\mathcal{C}}$ of size n.

2) Put *all* training cases ($\boldsymbol{\mathcal{O}}_i \cup \boldsymbol{\mathcal{O}}_{-i}$) down each tree. If cases C_i and C_j are in the same terminal node, then $U_{ij}^m = 1$; otherwise $U_{ij}^m = 0$.

3) Let $U_{ij} = \frac{1}{M}\sum_{m=1}^{M} U_{ij}^m$. Then the proximity for $\boldsymbol{\mathcal{C}}$ based on this RF is a symetric $n \times n$ matrix $\boldsymbol{P}$, with its ith row and jth column entry equal to U_{ij}, which is the proximity between C_i and C_j.

This proximity matrix is provided by the R package *randomForest*.

5.4 Predictor Variable Importance

There is no universally accepted criterion of predictor variable importance. One natural notion is that a predictor variable is important if prediction performance can be improved by using the predictor and diminished by ignoring it (Olshen 2010, pp. 1644). In Chapter 2 we will show this definition can be misleading.

Breiman proposed another way to define variable importance (Breiman 2001b) as follows. For a tree, if randomly permuting the values of a certain predictor variable for cases not included in tree construction harms the prediction for those cases, then this variable is deemed important. The variable importance defined in this way assigns highest values to variables with the greatest discrepancy between original prediction performance and prediction performance after permutation. The resulting variable importance values have no meaning on an absolute scale, but their relative sizes can be useful for comparing across different predictor variables. In regression problems, such prediction performance is computed by first tree-wise determining squared errors for predicting training cases left-out of the resampled tree construction and then averaging all such errors over all trees in the forest. In classification problems, increase in misclassification error after permutation can be taken as the measure of variable importance. This notion is embodied in the variable importance measure of the R package *randomForest*.

5.5 Predictor Variables with Missing Values

Breiman suggested two ways of imputing missing values in predictor variables when using RFs for prediction (see Breiman's homepage). Notationally, suppose the jth predictor from case C_i, x_{ij} is missing. For classification problems, if X_j is a numerical variable, a simple approach is to replace this missing value with the median of all available jth predictor values from cases with observed class y_i. If X_j is a categorical predictor variable, replace the missing value with the most frequent category observed in all cases with non-missing values of the predictor and observed class y_i.

Missing values in regression problems (those with quantitative response variable) is easier to deal with than classification problems. A simple imputation approach is to replace the missing value with the median of all available jth predictor values from the training set $\boldsymbol{\mathcal{C}}$, regardless of their response values.

A more complicated iterative method is as follows.

1) Use the simpler approach described above to obtain starting values for missing observations.

2) Grow a RF based on the imputed training set, and calculate all pairwise proximity measures as discussed in Section 5.3.

3) The missing value on jth predictor for C_i, x_{ij}, is replaced with a weighted average of non-missing values based on proximity measures M_{ik}, $k \in \mathcal{Q}_j$, where $\mathcal{Q}_j$ is the subset of the training set whose jth predictor values are not missing, i.e.,

$$\tilde{x}_{ij} \quad = \quad \frac{\sum_{k \in \mathcal{Q}_j} M_{ik} \cdot x_{kj}}{\sum_{k \in \mathcal{Q}_j} M_{ik}} \, .$$

4) Repeat steps 2 and 3 iteratively.

The second imputation method is more computationally intensive but provides better prediction performance. Both methods are available in the R package *randomForest*.

6 Random Forest Variations

Since the invention of Breiman's RF methodology, several variations on random forests have been developed, including random survival forests (RSF, Ishwaran et al. 2008), multivariate random forests (De'ath; Segal and Xiao 2011), enriched random forests (Amaratunga et al. 2008), quantile regression forests (Meinshausen 2006), etc. These methods either deal with different problem contexts, or were proposed with different resampling or response surface fitting mechanisms, which all contribute to the generalization of RF methodology.

6.1 Random Survival Forests

Standard RFs cannot deal with survival data with censoring. Ishwaran et al. proposed growing decision trees with bootstrap resamples based on a splitting rule maximizing the logrank statistic after splits (Ishwaran et al. 2008). Survival functions are estimated non-parametrically in terminal nodes. The authors also provided their computation tool in R, called *randomSurvivalForest*. In this package, they used the Nelson-Aalen method to estimate survival functions.

6.2 Multivariate Random Forests

De'ath (2010) considered using the idea of decision trees to handle regression problems with multivariate responses. Instead of measuring the sum of squared errors (SSEs) before and after splits on one response, the author used the sum of the SSEs from multivariate responses as the measure of node heterogeneity. Therefore, the chosen split is the one giving the maximal decrease in the sum of SSEs. The computation tool was provided by the same author. Segal and Xiao (2011) further proposed growing an ensemble of these multivariate regression trees.

6.3 Enriched Random Forests

When the number of predictors is huge yet the number of informative predictors is relatively very small, the classification performance by RF declines dramatically because a RF randomly selects a subset of features as splitting candidates at each split. This situation can be met, for example, in microarray analyses that typically involve thousands of genes whose expression levels serve as predictor variables. Amaratunga et al. (2008) suggested evaluating how informative each predictor variable is using some pre-filtering methods, such as t-tests, in order to roughly calculate how well a predictor can separate two classes. Then the random selection of predictor variables is replaced by weighted selection based on the pre-filtering procedure. Thus, the chance of selecting informative variables is largely increased.

6.4 Quantile Regression Forests

RFs take the average of predictions by individual trees as the predicted value by the forest. Meinshausen (2006) argued that a RF actually contains the estimated distribution of $F_{Y|\boldsymbol{x}}(y \mid \boldsymbol{X}_0 = \boldsymbol{x})$, not only the mean $E(Y \mid \boldsymbol{X}_0 = \boldsymbol{x})$. In RFs, only the averages of responses in terminal nodes are recorded in a tree. In contrast, quantile regression forests save all training case indices in terminal nodes. Note that the predicted value by a RF is nothing but a weighted average of the training sample responses. The authors derived the estimator for $F_{Y|\boldsymbol{x}}(y \mid \boldsymbol{X}_0 = \boldsymbol{x})$ using a weighted average of $\mathcal{I}(Y_i \leq y)$, where $\mathcal{I}(\cdot)$ is

the indicator function and Y_i is the response of training case C_i. Consistency of this estimator was proven under certain assumptions.

References

Amaratunga, D., Cabrera, J., and Lee, Y.-S. (2008). Enriched random forests. *Bioinformatics*, 24(18):2010–2014.

Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588.

Borra, S. and Ciaccio, A. D. (2010). Measuring the prediction error. a comparison of cross-validation, bootstrap and covariance penalty methods. *Computational Statistics and Data Analysis*, 54:2976–2989.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, 26(2):123–140.

Breiman, L. (1996b). Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, 24(6):2350–2383.

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

Breiman, L. (2001b). Statistical modeling: The two cultures (with rejoinder). *Statistical Science*, 16(3):199–231.

Breiman, L., Friedman, J., Stone, C., and Olshen, R. (1984). *Classification and Regression Trees.* Chapman and Hall, 1st edition edition.

Buhlmann, P. and Yu, B. (2002). Analyzing bagging. *Annals of Statistics*, 30(4):927–961.

Coppersmith, D., Hong, S., and Hosking, J. (1999). Partitioning nominal attributes in decision trees. *Data Mining and Knowledge Discovery*, 3:197–217.

De'ath, G. mvpart: Multivariate partitioning, 1.3-1. *R Package Version*.

Diaz-Uriarte, R. and de AndršŠs, S. A. (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7:3–15.

Efron, B. (2004). The estimation of prediction error: Covariance penalties and cross-validation. *Journal of the American Statistical Association*, 99(467):619–642.

Efron, B. and Tibshirani, R. (1997). Improvements on cross-validation: The .632+ bootstrap method. *Journal of the American Statistical Association*, 92(438):548–555.

Freund, Y. and Schapire, R. (1996). Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156.

Friedman, J. and Hall, P. (2000). On bagging and nonlinear estimation. *Journal of Statistical Planning and Inference*, 137(3):669–683.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63:3–42.

Goldstein, B., Polley, E., and Briggs, F. (2011). Random forests for genetic association studies. *Statistical Applications in Genetics and Molecular Biology*, 10(1):1–34.

Hammann, F., Gutmann, H., Vogt, N., Helma, C., and Drewe, J. (2010). Prediction of adverse drug reactions using decision tree modeling. *Clinical Pharmacology and Therapeutics*, 88:52–59.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2nd edition edition.

Ho, T. (1995). Random decision forest. *Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC*, 14-16:278–282.

Ho, T. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844.

Ishwaran, H., Kogalur, U., Blackstone, E., and Lauer, M. (2008). Random survival forests. *The Annals of Applied Statistics*, 2(3):841–860.

Jiang, P., Wu, H., Wang, W., Ma, W., Sun, X., and Lu, Z. (2007). Mipred: classification of real and pseudo microrna precursors using random forest prediction model with combined features. *Nucleic Acids Research*, 35(2):339–344.

Kumar, M. and Thenmozhi, M. (2006). Forecasting stock index movement: A comparison of support vector machines and random forest. *Indian Institute of Capital Markets 9th Capital Markets Conference.*

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

Lin, Y. and Jeon, Y. (2006). Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590.

Meinshausen, N. (2006). Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999.

Olshen, R. (2010). Remembering leo breiman. *The Annals of Applied Statistics*, 4(4):1644–1648.

Pal, M. (2003). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222.

Palmer, D., O'Boyle, N., Glen, R., and Mitchell, J. (2007). Random forest models to predict aqueous solubility. *J Chem Inf Model*, 47(1):150–158.

Segal, M. and Xiao, Y. (2011). Multivariate random forests. *WIREs Data Mining and Knowledge Discovery*, 1:80–87.

Shi, T., Seligson, D., Belldegrun, A., Palotie, A., and Horvath, S. (2005). Tumor classification by tissue microarray profiling: Random forest clustering applied to renal cell carcinoma. *Modern Pathology*, 18(4):547–557.

Siroky, D. (2009). Navigating random forests and related advances in algorithmic modeling. *Statistics Surveys*, 3:147–163.

Ward, M., Pajevic, S., Dreyfuss, J., and Malley, J. (2006). Short-term prediction of mortality in patients with systemic lupus erythematosus: Classification of outcomes using random forests. *Arthritis and Rheumatism*, 55:74–80.

CHAPTER 2. PERDICTOR AUGMENTATION IN RANDOM FORESTS

Ruo Xu, Dan Nettleton, and Daniel J. Nordman

Ruo Xu is Graduate Student, Department of Statistics, Iowa State University, Ames IA 50011 (Email: xuruo@iastate.edu); Dan Nettleton is Professor, Department of Statistics, Iowa State University, Ames, IA 50011 (Email: dnett@iastate.edu); Daniel J. Nordman is Associate Professor, Department of Statistics, Iowa State University, Ames, IA 50011 (Email: dnordman@iastate.edu). This work was supported by National Science Foundation-Plant Genome Award 0922746.

Abstract

Random forest (RF) methodology is a nonparametric methodology for prediction in both regression and classification problems. In this paper we describe a perhaps unexpected behavior of random forests (RFs): out-of-sample prediction by RFs can be sometimes improved by augmenting the design matrix with a new explanatory variable, independent of all variables from the original dataset. We explain this phenomenon with a simulated example, and conclude that independent variable augmentation helps RFs to spread weights on training samples when predicting a test case response, which decreases the prediction variance and may possibly improve prediction performance. We also give real data examples for illustration, and argue that this phenomenon is closely connected with overfitting. Because RFs have been criticized for being difficult to analyze directly, we also aim to present the inner-workings of RFs from a helpful perspective (related to weight-spreading in predictions as weighted averages of training responses), and to further suggest potential research for improving RFs.

Keywords: Regression; Classification; Machine learning.

1 Introduction

Random forest (RF) methodology is among many machine learning techniques useful for prediction and classification problems (Breiman 2001a). The popularity of random forests (RFs) is reflected by its extension and incorporation in other methodology, such as multivariate random forests (De'ath; Segal and Xiao), quantile regression forests (Meinshausen 2006), enriched random forests for microarray analysis (Amaratunga et al. 2008), random survival forests (Ishwaran et al. 2008) and the R package "*pathwayRF*" for metabolic pathway analysis, etc. The RF approach has also been found to work well in high dimensional problems (Breiman 2001b).

A RF is a collection of classification or regression trees generated by a bootstrap procedure. Each tree is grown from an independent bootstrap resample until all nodes contain observations no more than a pre-specified maximal node size. No pruning is done, unlike in the case of a single tree. Each tree in the forest then provides a prediction of a response variable of interest, and a single overall prediction is obtained by taking a weighted average over the tree predictions from the forest. In a regression problem, equal weights are assigned to every tree (i.e., sample average); in a classification problem, the class predicted by the most trees is taken as the prediction. This numerical approach of growing trees in a forest through a series of bootstrap resamples and creating predictions as their averages is typically how the RF method is implemented in practice, per Breiman's (2001a) original intention. However, more formally, predictions from the RF method are statistics defined by a bootstrap expectation (i.e., a well-defined expected value from a bootstrap mechanism applied to re-creating a data functional corresponding to *trees*), and the numerical implementation of growing trees provides a Monte-Carlo approximation to this well-defined, but difficult to directly compute, bootstrap expectation. In this sense, the RF methodology is a direct application of so-called bagging (bootstrap aggregation) to trees. The RF method has sometimes been referred to as a "black box" because its properties are difficult to study analytically. While some statistical progress has been made in clarifying RFs, particularly as a nearest neighbor method (Lin and Jeon 2006; Hastie et al. 2009, Chapter 15), translating and interpreting the general mechanics of RFs for prediction remains challenging.

The purpose of this paper is to investigate a curious and somewhat counter-intuitive feature of RFs that we have encountered in our use of the method; namely, *out-of-sample* predictions can often be *improved* by augmenting an original dataset with random explanatory (or predictor) variables, created independently of all variables in the original dataset. This is entirely different from the standard linear regression scenario where better in-sample predictions (e.g., higher R^2 values) can be obtained by simply including additional predictor variables. In that case, including random or meaningless explanatory variables usually leads to

higher mean squared prediction error by increasing variation without reducing bias. However for RFs, perhaps surprisingly, this type of data augmentation can induce smaller mean squared errors (MSEs) in regression, and lower misclassification rates in classification, when predicting outside of the data used to develop the forest.

We introduce this phenomenon with an example involving simple regression. Consider a training sample of 100 iid observational pairs (X_1, Y) with $X_1 \sim U(0,1)$ and $Y|X_1 \sim N(X_1, 0.3^2)$. Suppose that an independent test case is drawn from the joint distribution of (X_1, Y), and that we wish to predict the response Y using knowledge of its X_1 value and a RF built from the 100 training cases. Applying the RF methodology, without using knowledge of the joint distribution of (X_1, Y), gives a prediction MSE of 0.132 (approximated from 1000 simulations). When repeating this entire process with datasets obtained by augmenting (X_1, Y) with a second predictor variable $X_2 \sim U(0,1)$, generated independently of (X_1, Y), the RF method using both X_1 and X_2 as predictors interestingly provides approximately a 12% reduction in MSE compared to the RF method without X_2.

The improvement in prediction by augmenting a dataset with an independent predictor is not limited to regression problems. As an example that a similar phenomenon can occur in classification problems, consider a binary response variable $Z = I(Y > 0.5)$ defined in the context of our previous example. When predicting the class of Z (0 or 1) using a RF, the average correct classification rate approximated from 1000 simulations increased from 68% to 73% when the single predictor X_1 was augmented with the independent predictor X_2.

We have found that it is possible to obtain better test case prediction by augmenting real datasets with independent random predictors. Because better predictions can result from including explanatory variables unrelated to the response, not all variables that improve RF predictions should be regarded as important. While no universally accepted criterion of variable importance exists, several researchers, including Breiman (2001b, pp. 229-30), have suggested that a predictor variable is important if prediction performance can be improved by its presence and harmed by its absence (Olshen 2010, pp. 1644). Our results indicate that this variable importance criterion may be misleading at times when RFs are used. Furthermore, the fact that such augmentation can improve RF predictions also suggests potential for further improvements to RF methodology.

The structure of this paper is as follows. In Section 2, we explain the reasons behind the, perhaps unexpected, improvement of RFs by predictor variable augmentation. We tie this feature to the weight selection used by RFs to obtain weighted averages of training responses and, more specifically, to the issues of weight-spreading and overfitting. In Section 3 we provide two real data examples, one for regression and one for classification, to illustrate the effect of predictor augmentation in analyses by RFs. In Section

4, we discuss some implications and generalizations of our findings and connect our work to some other known results about RFs (Lin and Jeon 2006). Section 5 provides some concluding and qualifying remarks about data augmentation in RFs.

It is well known, and will be explained in the following, that introducing certain types of randomness into the RF *procedure* may improve predictions (e.g., defining each tree from a bootstrap resample or defining a split in a tree by a random selection of only a subset of predictor variables, cf. Breiman 2001a). The phenomenon that we discuss here is different. We are not altering the RF procedure itself, but rather examining the impact of augmenting an initial data set with meaningless predictor variables. It is statistically counterintuitive that such augmentation can improve the performance of a serious prediction method, but nonetheless, such improvements are possible with random forests. Although we demonstrate the phenomenon of prediction improvement with augmentation, we do not attempt to develop a new machine learning method (i.e., alter the RF procedure), nor do we aim to suggest a practical way to improve the prediction performance by RFs. Instead, by illustrating an effect of data augmentation, we hope to explain RF mechanics from a different perspective and again suggest some caution in interpreting predictor variables which improve RF predictions.

2 Improved Predictions via Data Augmentation with Independent Explanatory Variables

To explain the improvements to RFs by independent predictor augmentation as alluded to in Section 1, we conducted a more elaborate simulation study with 1000 simulation runs, where each run used the following data-generating procedure. In each particular simulation, we generated a training sample with 101 cases (X_1, Y), created from a non-random explanatory variable, equally partitioning the interval $[0, 1]$ as $X_1 = i/100, i = 0, \ldots, 100$, and a response variable Y generated as $Y|X_1 \sim N(X_1, 0.3^2)$. This **o**riginal dataset will be referred to as *dataset-O*. For the purpose of comparison, we created another dataset by **a**ugmenting dataset-O with an independent variable $X_2 \sim U(0, 1)$ in the same simulation run. We refer to this second dataset as *dataset-A*. Two RFs were grown from the datasets with or without the X_2 variable (denoted as RF-A and RF-O, respectively); each forest had 100 fully grown trees with a maximal node size of 1. Note that the two datasets in each simulation run shared exactly the same X_1 values and Y values, with the only difference between the datasets being whether or not X_2 was present. Additionally, a tree in RF-O was grown using the same bootstrap index as a corresponding tree in RF-A, which helped to reduce variability in the simulation study induced by bootstrap resampling. We used RF-O and RF-A to predict responses at four values of $X_1 = 0$, 0.25, 0.5, 0.75. To evaluate prediction performance, within

the same simulation run, we generated 10 independent response cases at each X_1 value and, separately for each X_1 value, computed the average squared prediction errors and biases between the RF predictions and actual test responses. Averaging these (average) prediction errors and biases produced the MSEs and estimated biases listed in Table 1. As Table 1 shows, augmenting with an irrelevant explanatory variable $X_2 \sim U(0,1)$ reduced the prediction MSE at each of the four test sample X_1 values. The improvement was more obvious for the predictions with an X_1 value in the middle of this variable's range $[0,1]$, rather than at the edges.

Table 1: Predictions by the RFs grown from the original (RF-O) and augmented (RF-A) datasets.

Test case X_1 values	Prediction MSE		Estimated Bias	
	RF-O	RF-A	RF-O	RF-A
$X_1 = 0$	0.1323	0.1189	0.008	0.078
$X_1 = 0.25$	0.1328	0.1112	0.003	0.017
$X_1 = 0.5$	0.1320	0.1073	-0.001	-0.003
$X_1 = 0.75$	0.1287	0.1081	-0.003	-0.012

To begin to understand the results in Table 1, we recall that, as mentioned in Section 1 for the regression case, a prediction by a RF amounts to an average of the predictions produced by the trees in the forest, where each tree is grown from an independent bootstrap resample of the original data. Because each tree prediction corresponds to some average of the responses $Y_1, \ldots, Y_n$ observed in the original training data (i.e., the average of responses in a node from the dataset used to grow the tree), we can view the final prediction of the RF (at some given level of explanatory variables $\boldsymbol{x}_0$) as a convex combination of the training responses

$$\hat{Y}(\boldsymbol{x}_0) = \sum_{i=1}^{n} w_i(\boldsymbol{x}_0) Y_i \tag{8}$$

involving nonnegative weights $w_i \equiv w_i(\boldsymbol{x}_0)$ with $\sum_{i=1}^{n} w_i = 1$. The weights w_i are functions of the training sample and the regressor value of the test case $\boldsymbol{x}_0$. A single tree in the forest is grown by a series of partitions of the regressor space (i.e., binary splits), which tend to pool data cases with similar regressors in the same nodes. As a result, a RF predicts a new case by selecting training cases over each tree that are close in terms of the explanatory variables, essentially producing a weighting scheme $w_1, \ldots, w_n$ that attempts to put more weight on responses $Y_1, \ldots, Y_n$ in the training dataset with explanatory variables that match those at which a prediction is desired. RFs created with or without augmentation predictor variables (i.e., RF-A or -O) are attempting to produce weights that achieve a good prediction of the response $Y_0 \equiv Y_0(\boldsymbol{x}_0)$ at some given level of the regressors. Letting $\boldsymbol{w} = (w_1, \ldots, w_n)^T$ and $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^T$, the quality of a predictor $\hat{Y} \equiv \hat{Y}(\boldsymbol{x}_0) = \sum_{i=1}^{n} w_i Y_i = \boldsymbol{w}^{\boldsymbol{T}} \boldsymbol{Y}$ of an independent

response Y_0 in terms of MSE, given by

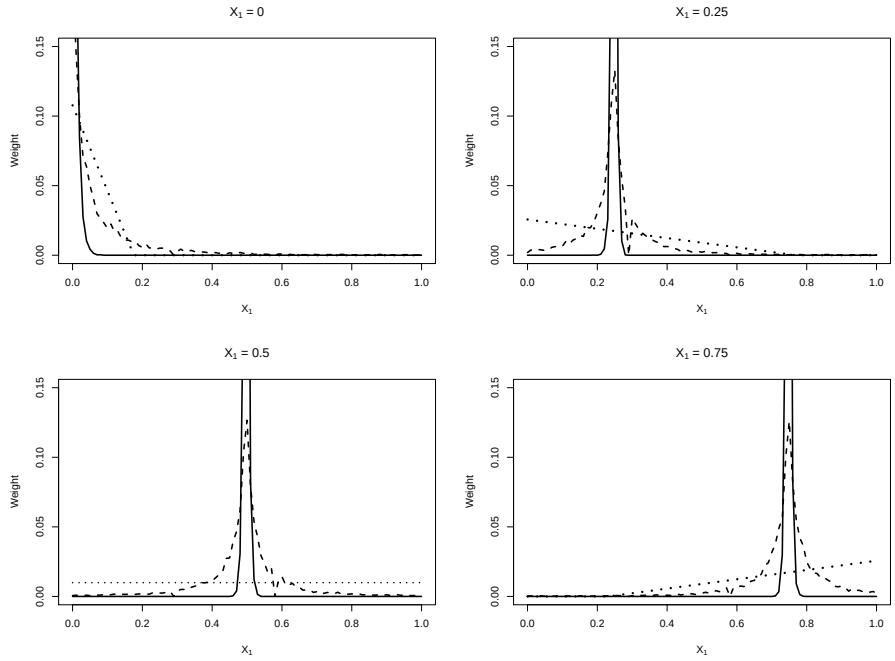
$$\begin{aligned} E(Y_0 - \hat{Y})^2 &= Var(Y_0) + Var(\hat{Y}) + [E(\hat{Y}) - E(Y_0)]^2 \\ &= Var(Y_0) + Var(\boldsymbol{w}^T\boldsymbol{Y}) + [E(\boldsymbol{w}^T\boldsymbol{Y}) - E(Y_0)]^2, \end{aligned} \tag{9}$$

depends on the weights $\boldsymbol{w}$ through the variance $Var(\hat{Y})$ and bias $E(\hat{Y}) - E(Y_0)$ of the predictor $\hat{Y} = \boldsymbol{w}^T\boldsymbol{Y}$. As in other regression problems, a trade-off exists in the RF method between prediction bias and variance, which are induced in this case by the selection of weights.

For the same simulation study that produced the prediction MSEs in Table 1, we can closely examine the weights (8) assigned to the training cases in a RF construction. Recall that each of the 101 training observations in this study corresponds to a unique value of a non-random explanatory variable $X_1 = i/100, i = 0, \ldots, 100$, chosen to equally partition the interval [0, 1]. Hence, from 1000 simulation runs, we determined the average value of a weight w_i assigned by a RF to a response Y_i corresponding to explanatory variable $X_1 = i/100, i = 0, \ldots, 100$, when trying to predict a new response Y generated at each of the X_1 levels 0, 0.25, 0.5, and 0.75 listed in Table 1. The results are displayed in Figure 1 for both RF-O and RF-A. Again, this figure gives an idea of how RFs with or without augmentation variables tend to select and weight training cases that are close in the regressor space to the positions at which predictions are desired. Also included in Figure 1 for comparison are the optimal weights $w_0, \ldots, w_{100} \geq 0$ with $\sum_{i=0}^{100} w_i = 1$ which minimize the prediction MSE (9); these values were computed numerically based on knowledge of the true mean response $E(Y|X_1)$ and variance σ^2, so such weights could not be used in practice. However, optimal weights are useful for comparison against RF weighting schemes.

Figure 1 illustrates the reason for the improvement to RF by independent predictor variable augmentation. When predicting a test case, RF-O tended to concentrate weights only on a few training cases with X_1 values immediately neighboring the X_1 value of the test case; in contrast, RF-A tended to spread nonzero weights on more training cases. This often led to slightly more bias but substantially less variance for RF-A predictions. For predictions of a new response at X_1=0, RF-A clearly led to more bias (see Table 1) because the weights on training cases could be only spread to training cases with a mean response greater than the mean response at X_1=0. But, in this study, bias cost much less than the gains made in increased precision, and hence a uniformly smaller MSE was obtained by RF-A. In Figure 1, the inclusion of X_2 dragged the weight assignment in RF-A from the narrow assignment of RF-O towards the optimal one. This weight-spreading effect helped to incorporate more training cases that were appropriately close to a test case (in terms of the meaningful regressor X_1) when forming a weighted average prediction.

Figure 2: Average weights on responses in the training set (identified by their X_1 values) that contribute to a prediction by either RF-O (–) or RF-A (- -). The dotted line ($\cdots$) corresponds to the weights used by the best linear predictor of the form (8) that minimizes prediction mean squared error given in (9).



We also tried augmenting the dataset with different numbers of independent $U(0, 1)$ predictors. When the number of irrelevant predictor variables was not very large (2 to 5), the prediction MSE for RF-A was smaller than for RF-O for test cases with X_1 values away from the 0 or 1 edges of this variable's range, but the degree of improvement was less than that shown in Table 1. This further augmentation had a less substantial effect in reducing the variance of predictions, as balanced against the corresponding losses in predictor accuracy. Furthermore, as the number of irrelevant predictors further increased (e.g., larger than five), the weights on responses in the training sample used for prediction became increasingly uniform. In these cases, bias increased and overwhelmed potential reductions in variance, leading to poor prediction performance except for predictions at X_1 values near the center of the [0, 1] interval, where a uniform weighting is optimal (results not shown).

To further describe how augmenting the design matrix affects weight-spreading in RF predictions, it is helpful to provide some details on how trees in the forest are grown. The description provided here is consistent with the default settings of the R package *randomForest* (Liaw and Wiener 2002). In the RF procedure, a bootstrap resample of the training sample is used to grow a tree by a series of node splits, where to determine a node split, the algorithm considers a series of binary partitions composed from a set of randomly chosen predictor variables and their values. Consistent with Breiman (2001a), the

number of randomly selected predictors considered for each split is $\max\{1, \lfloor p/3 \rfloor\}$ for regression, where p is the total number of predictors. A node split is selected from the midpoints of the intervals between ordered values from the randomly selected predictors (from a set of available cases), and two nodes are subsequently defined by splitting the values from the predictor variable into two disjoint intervals along a selected midpoint.

As an example, suppose a node is to be split, consisting of three training cases with two explanatory variables: $C_1(X_1 = 0, X_2 = 0)$, $C_2(X_1 = 1, X_2 = 1)$, and $C_3(X_1 = 2, X_2 = 2)$. If we randomly select the X_1 variable for the split, then we would have two possible partitions, $(-\infty, 0.5] \cup (0.5, \infty)$ or $(-\infty, 1.5] \cup (1.5, \infty)$, depending on the midpoint selected for the split. The mean surface for each node/partition is estimated by the average response values over training cases in the node. The tree growing procedure is a greedy one, in that it always chooses the current best partition (midpoint for the split) in each step in order to minimize some loss criterion (i.e., squared error loss in regression), though this immediate best partition may not be the globally best one.

Consider the simulation of the last section, with two predictor variables (X_2 being the augmentation variable). When growing both RF-O and RF-A, only one predictor (either X_1 or X_2) is randomly selected for consideration at each split. Because the trees in RF-O construction must split only on the right predictor X_1, a *narrow* weighting scheme is induced, and the training cases ultimately weighted to predict a new test case are very close to each other in terms of the distance between their X_1 values. On the other hand, with RF-A construction, both X_1 and X_2 have equal chances to be considered at a split. Splits based on the noninformative X_2 variable may often direct the test case to a node that does not contain the training cases closest to the test case in terms of distance between their X_1 values. Subsequent splits based on X_1 will tend to place the test case with training cases with similar X_1 values, among those training cases that have not already been split onto different branches of the tree. In this way, splits on the noninformative X_2 variable tend to diversify the tree structures and spread weights on more training cases in prediction, which are roughly close to a new test case in terms of X_1 values (even though splits on the variable X_2 are not necessarily meaningful).

As an extremely simple illustration, suppose there are two training cases, $C_1 \equiv C_1(Y_1, X_1 = 0, X_2 = x_{21})$ and $C_2 \equiv C_2(Y_2, X_1 = 1, X_2 = x_{22})$ available for predicting an independent test case $(Y_3, X_1 = 0, X_2 = x_{23})$, where x_{21}, x_{22}, and x_{23} are realizations of iid $U(0,1)$ random variables, and $Y|X_1 \sim N(X_1, \sigma^2)$, similar to our previous simulation example. Consider three different possibilities for predicting the test case response.

(a) A tree is grown on the right variable X_1, with neither bagging nor predictor augmentation, which produces a prediction Y_1 with prediction MSE (9) given by $2\sigma^2$; here the prediction bias is zero and the

prediction variance is $2\sigma^2$.

(b) The RF-O prediction (i.e., using only X_1) as a bootstrap expectation, based on bagging trees grown from the following resamples $\{C_1, C_1\}$, $\{C_2, C_2\}$ and $\{C_1, C_2\}$ (with probability 1/4, 1/4, and 1/2), is $3Y_1/4 + Y_2/4$ with prediction MSE given by $4/64 + 104\sigma^2/64$, where the first and second terms in the sum correspond to squared prediction bias and prediction variance, respectively. Note also here that the RF prediction, as a bootstrap expectation, was computed directly, without numerical approximation involving resampled trees.

(c) The RF-A prediction as a bootstrap expectation, based on bagging the same resampled trees as RF-O but with X_2 augmentation, is given by $Y_1/4 + Y_2/4 + M/2$, where Y_1, Y_2 and M are the outputted predictions from resamples $\{C_1, C_1\}$, $\{C_2, C_2\}$ and $\{C_1, C_2\}$, respectively, with M defined as

$$M = \frac{1}{2}Y_1 + \frac{1}{2}\begin{cases} Y_1 & \text{if } x_{23} \leq (x_{21} + x_{22})/2 \text{ and } x_{21} \leq x_{22} \\ Y_2 & \text{if } x_{23} \leq (x_{21} + x_{22})/2 \text{ and } x_{21} > x_{22} \\ Y_1 & \text{if } x_{23} > (x_{21} + x_{22})/2 \text{ and } x_{21} > x_{22} \\ Y_2 & \text{if } x_{23} > (x_{21} + x_{22})/2 \text{ and } x_{21} \leq x_{22}. \end{cases}$$

The dichotomous definition of M owes to the fact that resample $\{C_1, C_2\}$ offers two responses as potential tree output and a tree is built from one regressor variable, randomly chosen between X_1 and X_2. For this example, with respect to the iid $U(0,1)$ draws defining the X_2 variable, we would expect M to be $3Y_1/4 + Y_2/4$ (interestingly *matching* the RF-O prediction in (b)), and we would then expect the final RF-A prediction to be $5Y_1/8 + 3Y_2/8$, with MSE given by $9/64 + 98\sigma^2/64$ (again a two part sum consisting of squared biases and prediction variance).

We see that the RF procedure itself (RF-O) assigns more weight on Y_2, when compared to a single tree (no bagging), because bagging leads to unavailability of C_1 in some resamples. Augmenting the design matrix with an independent $U(0,1)$ variable X_2 further spreads the weights from Y_1 to Y_2, because the possibility of splitting on X_2 increases the potential for Y_2 to be used for prediction even when C_1 is in the bootstrap resample. This example intuitively explains how predictor augmentation helps to spread weights on training samples in prediction. As the noise level σ^2 increases, the RF-O method becomes preferred (with respect to MSE) over a single tree, and RF with augmentation becomes preferred over RF-O. More specifically, in this simple example, (a) has lowest MSE for $\sigma^2 < 1/6$, (b) has lowest MSE for $\sigma^2 \in (1/6, 5/6)$, and (c) has lowest MSE for $\sigma^2 > 5/6$. Note that we do compromise prediction accuracy (larger squared prediction bias) by implementing RF and additionally by predictor augmentation, but the pay-off in reduced prediction variances improves the overall MSE for larger σ^2. We will discuss this

issue in Subsection 4.2.

From the above argument, we also see that the weighting scheme of training samples used for prediction crucially influences the prediction MSE, and this perspective also largely explains the prediction improvements made by RFs over single trees in the first place. When the maximal node size is one, a tree typically predicts a test case using only one training case, and corresponds to the narrowest possible weighting scheme of the training responses. Bagging then helps to broaden the spread of weights in a convex combination of training responses as a predictor (8) based on training responses as long as the number of *structurally different* trees is not too small. In a sense, this diversification of weights provides a different, though related, perspective for understanding Breiman's (2001a) original argument that RFs improve prediction by combining trees that are made less "correlated" (in Breiman's words), or more diverse, by randomly selecting subsets of predictors (instead of using them all) to build trees. From a perspective of weight-spreading, similarly structured trees, even if built by resampling, will tend to narrowly focus weights on a few training responses, while the prediction by more structurally diversified trees (i.e., diversified by randomly choosing regressor variables in resampled trees and, in our case, further diversified by predictor variable augmentation) tend to positively weight a wider range of appropriate training cases and thereby reduce prediction variance. This again intuitively explains how RFs alleviate overfitting compared to an individual tree.

In the standard Monte Carlo (MC)-based implementation of a RF, where we numerically determine the RF prediction from a group of trees grown by a finite set of bootstrap resamples, we also note that there is a separate issue of using a sufficient number of trees (i.e., bootstrap resamples) to obtain a reasonable MC approximation, and the number of tree resamples can also impact the final weights used in a RF prediction in practice. However, simply increasing the number of resamples (or trees in a MC-constructed forest) only improves the MC-approximation to the weights assigned by a given RF procedure, and this does not change the structure of a RF-procedure itself (whether RF-O or RF-A). Our simulation study showed that the advantage of RF-A over RF-O in terms of prediction MSE remained unchanged when the number of resamples (trees per forest) increased from 100 to 10000 (results not shown). We will further discuss this issue of resample sizes in Sections 3 and 4.

3 Improvement by Variable Augmentation in Real Data Examples

While the previous section considered simulated data, improving test sample prediction performance of a RF by augmenting the design matrix with independent explanatory variables also occurs in real data analyses. For illustration, we first present some results based on the concrete compressive strength data of Yeh (1998). The dataset has 1030 observations, with eight quantitative input variables and a response variable, concrete compressive strength. We performed regression analyses (predictions) by RFs based on the original and augmented datasets, and we also examined the effect of maximal node size in the trees. We created 1000 independent partitions of the original data, where each time we randomly divided the data into a training set (with 1000 observations) and a test set (the remaining 30 observations) and augmented the original design matrix with an independent $U(0, 1)$ predictor variable as in Section 2. RF-A and RF-O were both grown with the maximal node size 1, 5, and 10 using the same 1000 training cases. The performance was evaluated based on prediction MSE of the test samples, averaging over test samples across the 1000 generated partitions. The results in Table 2 indicate that predictor augmentation reduced the prediction MSE, regardless of node size and number of trees in a RF. In this example, growing each tree to its largest possible form (maximal node size 1) produced better predictions than the default setting (maximal node size 5) in the R package *randomForest*.

Table 2: Prediction MSEs in the concrete compressive strength regression with (RF-A) and without (RF-O) predictor augmentation.

Maximal node size	Prediction MSE					
	RF-O			RF-A		
	1	5	10	1	5	10
10 trees/resamples used	35.87	37.88	42.48	34.78	36.16	39.02
100 trees/resamples used	28.50	30.97	35.74	27.81	29.34	32.96
1000 trees/resamples used	28.42	30.98	35.73	27.67	29.29	32.94

Breiman (2001a) claimed that RFs do not overfit by showing its generalization error for a classification problem converges for an infinite number of trees. This suggests that increasing the number of trees in a RF is enough to solve the overfitting problem in RFs. Lin and Jeon (2006) showed that controlling node size improved prediction performance by RFs in some datasets. Table 2 shows that the relative performance of RF-O could not be improved either by increasing the number of trees per forest (resamples in the implementation of a RF), or increasing node size. The effect of node size and number of trees per RF will be further discussed in Subsection 4.1

We also tested predictor augmentation in a classification problem with a real data set. Haberman (1976) reported a dataset on the survival of patients who had undergone breast cancer surgery at the University of Chicago's Billings Hospital between 1958 and 1970. The dataset has three explanatory variables: age of patient at time of operation, year of operation, and number of positive axillary nodes detected. The binary response variable is patient's survival status five years after operation. There are 306 patients in this dataset. Again we augmented the original design matrix with an independent $U(0,1)$ random variable. The dataset was randomly partitioned into a training set with 2/3 of the patients and a test set with the remaining 1/3 of the patients. We grew both RF-O and RF-A with the same training set and predicted the response in the test set with the maximal node size 1, 5, and 10. This process was repeated 1000 times, and the average correct classification rates are shown in Table 3. Augmenting the design matrix with an independent $U(0,1)$ predictor improved classification in all scenarios. As in the previous regression problem with the concrete dataset, increasing the number of trees per forest led to little or no improvement. The default maximal node size of the *randomForest* package for classification problem is 1, which produced slightly worse predictions than the larger maximal node size considered for this dataset.

Table 3: Classification rates for Haberman's survival data with (RF-A) and without (RF-O) data augmentation.

Maximal node size	Correct classification rate					
	RF-O			RF-A		
	1	5	10	1	5	10
10 trees/resamples used	0.718	0.720	0.726	0.729	0.731	0.732
100 trees/resamples used	0.720	0.722	0.728	0.732	0.732	0.732
1000 trees/resamples used	0.721	0.723	0.728	0.732	0.731	0.732

4 Other Considerations Impacting the Effect of Variable Augmentation

In Subsections 4.1 through 4.3, we briefly connect the evidence of improved random forest (RF) predictions by predictor augmentation to several other aspects influencing the performance of RFs, such as the number and size of individual trees in a forest, signal-to-noise issues, the dimensionality of data, and the functional relationship between mean response and explanatory variables. We also return to the issue of interpreting variable importance in light of variable augmentation in Section .

4.1 Number and size of trees

As described in Section 2, a RF grows an ensemble of trees with bootstrap samples and thereby improves prediction (over a single, less stable tree, cf. Breiman et al., 1989) by averaging a series of tree structures, and inducing a broader set of weights on training responses. We have seen that predictor augmentation can further add to tree diversification and weight-spreading in a RF predictor (8) and that, when such augmentation is helpful, it is because this acts to reduce prediction variance and mitigate overfitting. However, the issue of overfitting by RF has been debated since the method's introduction. Breiman (2001a) claimed that RFs "never overfit" because RF generalization error for a classification problem converges as the number of trees increases. In our interpretation, this statement conveys that, as the number of resamples (and subsequent trees) increase in an MC-based implementation of RF, the numerical computation provides an increasingly better finite-sample approximation of the RF prediction, technically defined as a bootstrap expectation (cf. Section 1). This does not imply that RFs are immune to overfitting problems. Hastie et al. (2009) have shown that RFs can overfit despite increased tree numbers in an MC-implementation. Our predictor variable augmentation addresses a different issue than the number of trees in the RF construction, and we have illustrated that augmentation can improve predictions regardless of the number of trees.

Regarding tree size, Hastie et al. (2009) have also suggested that the overfitting of RFs with fully grown trees (i.e., maximal node size of 1) seldom costs much, especially in classification problems. It is commonly believed that RFs work best with a maximal node size of 1. However, the default maximal node size values in the R package *randomForest* are 5 for regression and 1 for classification problems. Segal (2004) and Lin and Jeon (2006) demonstrated minor gains in RF regression problems by controlling the node sizes of individual trees in a forest. In particular, Lin and Jeon (2006) related RFs to the adaptive k-nearest neighbor (kNN) method, and showed that tuning the maximal node size is advantageous for the performance of RFs. There can be an advantage of choosing a maximal node size larger than one for datasets with many observations but relatively small dimension, because this also has the effect of weight-spreading to reduce the variance of RF predictions. However, tuning the maximal node size of individual trees may not be enough to avoid overfitting problems in RFs completely, and predictor augmentation can still be beneficial with maximal node sizes larger than one. Our real data analysis examples in Section 3 (Table 2 and 3) indicate that prediction performance can be further improved by $U(0,1)$ predictor augmentation at different maximal node sizes. This suggests that, when such predictor augmentation is helpful, there may yet be room for improvement in RFs, including the choice of maximal node size.

4.2 Signal-to-noise issues and mean response function

The prediction variance for a given test case is, of course, related to the variance of the training data. More variability in the training data often translates into larger variances for RF predictions relative to their squared biases, and hence a larger benefit by predictor augmentation and widening the weights (8) on the training sample. In the simulation experiment of Table 1, for example, if we generate the response variable $Y|X_1$ from $N(X_1, 0.5^2)$ instead of $N(X_1, 0.3^2)$, the variance/MSE reducing effect is even more obvious by augmenting with an independent X_2 from $U(0, 1)$. In contrast, if $Y|X_1$ is from $N(X_1, 0.1^2)$, there is hardly any improvement. That is to say, data sets with large signal to noise ratios benefit less from predictor augmentation.

Our simulation example in Section 2 was admittedly and intentionally simple in that the mean function of response variable was linear in the only key predictor X_1 equally partitioning the unit interval $[0, 1]$. When the test sample has regressor values that are not too extreme so that there are training cases which approximately neighbor the test case in regressor space, averaging or weighting more training responses, induced by independent predictor augmentation, leads to better prediction precision without sacrificing too much prediction accuracy. However situations can arise where augmentation can hurt the overall prediction MSE because the gains in reduced prediction variance from weight-spreading cannot offset the damage in bias. This can occur, for instance, when the mean function is highly non-linear over small neighborhoods in the regressor space and the underlying noise is low. Consider another simple simulation study, similar to that in Section 2, where the mean function is $Y \sim \sin[N(X_1, \sigma^2)]$, $X_1 \sim U(0, 2\pi)$, and we examine the effect of augmenting with an independent $U(0, 1)$ predictor for predicting an independently drawn test case (Y_1, X_1). When σ is 0.3, RF-A and RF-O give prediction MSEs of 0.084 and 0.069, respectively. However, augmentation (weight-spreading) starts to help as the noise of responses rises, and when σ is 0.5, the prediction MSEs become 0.157 (RF-A) vs. 0.164 (RF-O). Prediction bias can be enlarged by the weight stretching effect of predictor augmentation. From Table 1, we see the estimated bias is larger for RF-A than for RF-O. It is possible that in some prediction problems, prediction bias introduced by augmentation might substantially increase if there are limited data available around the test case in the regressor space and if the underlying mean response curve sharply fluctuates over the regressor space.

4.3 Number of predictor variables

The number of predictor variables is another factor that affects the performance of a RF. Predictor augmentation is more effective if the predictor dimension is low. When the original dataset has many predictors, the weight-spreading effect by augmenting with independent predictors is weakened for two

reasons. First, the chance that a noninformative augmentation predictor will be selected decreases at each split. Second, when the original dataset has some irrelevant predictors already, the weights in (8) can be sufficiently spread by these irrelevant predictors, and an extra augmentation predictor may have little impact. For these reasons, we may not obtain smaller prediction error by augmenting a real dataset that already has many predictors.

However, it is often possible to select a subset of predictor variables and gain prediction improvement by predictor augmentation (in both regression and classification problems). For instance, in the survival classification problem on Haberman's dataset in Section 3, suppose we choose a design matrix with age of patient during operation and year of operation, but excluding the number of positive axillary nodes detected. For this reduced dataset, augmentation by an independent $U(0,1)$ predictor increases the correct classification rate from 0.673 to 0.733, about an 8% improvement (approximated by simulation), compared to the corresponding 1% improvement in Table 3. This aspect of RF performance has some implications for interpreting variable importance, as described in the next subsection.

4.4 Variable importance

Variable importance may have different meanings in different contexts, with no generally accepted definition. Often, a predictor variable may be regarded as important if the prediction on an independent test sample is more accurate with it, and is less accurate without it (the rejoinder of Breiman 2001b; Olshen 2010). Our examples demonstrate that, in RFs, the presence of predictors independent of the response variable may reduce the prediction error, even though such variables are obviously not important in any scientifically meaningful sense. This illustrates a possible pitfall with this definition of variable importance.

Breiman (2001b) also proposed a second way to define variable importance as follows: if, within each resampled tree, randomly permuting the values of a certain predictor variable harms the prediction for cases not included in the given tree construction, then this variable is deemed important. This notion is embodied in the "variable importance measure" of the R package *randomForest* (Liaw and Wiener 2002), which assigns highest values to variables with the greatest discrepancy between original prediction performance and prediction performance after permutation. (In regression problems, such prediction performance is computed by first tree-wise determining squared errors for predicting training cases left-out of the resampled tree construction and then averaging all such errors over all trees. The resulting variable importance values have no meaning on an absolute scale, but their relative sizes can be useful for comparing across different predictor variables). By its construction, this second variable importance measure can distinguish independent augmentation predictors from scientifically meaningful predictors

because permuting an independent augmentation variable has no impact on the joint distribution of the variables in the data set. Consequently, permuting an augmentation variable will not substantially change predictions and will typically result in a relatively low measure of variable importance. In the simple illustrative example we employed in Section 1 with $Y|X_1 \sim N(X_1, 0.3^2)$, X_1 and X_2 iid $\sim U(0,1)$, the variable importance measures given by *randomForest* (approximated from 1000 simulations) for X_1 and X_2 are 20.85 and 0.24, respectively. This correctly indicates that X_1 is far more important than the irrelevant variable X_2. Thus, Breiman's second criterion of variable importance is more meaningful than the first notion of variable importance when using RFs in variable selection and model building problems.

5 Conclusions and Qualifications

This paper demonstrated and investigated a peculiar phenomenon with RF methodology that independent predictor variable augmentation sometimes improves out-of-sample RF predictions. As part of this process, we provided a straightforward and intuitive explanation of how RFs work, and can be improved, as predictors in the form of convex combinations of training responses, through a process of weight-spreading. Augmenting a dataset with an independent predictor variable can often induce a type of weight-spreading which crucially reduces the variance of predictions compared to the additional bias induced, and thereby improve the prediction performance of RFs.

As part of this effort, we offer some warning that there is a potential risk in assuming that only useful explanatory variables will contribute to better RF prediction, because augmentation with scientifically meaningless variables demonstrates otherwise. To again qualify this work, our intention here is not to suggest or recommend predictor augmentation in practice for improving RFs. However, the fact that such data augmentation can improve RF predictions at all, despite maximal node size choices or numbers of resampled trees used in numerical construction of forests, indicates that there may exist further research potential to achieve better implementations of RF methodology in practice.

References

Amaratunga, D., Cabrera, J., and Lee, Y.-S. (2008). Enriched random forests. *Bioinformatics*, 24(18):2010–2014.

Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588.

Borra, S. and Ciaccio, A. D. (2010). Measuring the prediction error. a comparison of cross-validation, bootstrap and covariance penalty methods. *Computational Statistics and Data Analysis*, 54:2976–2989.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, 26(2):123–140.

Breiman, L. (1996b). Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, 24(6):2350–2383.

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

Breiman, L. (2001b). Statistical modeling: The two cultures (with rejoinder). *Statistical Science*, 16(3):199–231.

Breiman, L., Friedman, J., Stone, C., and Olshen, R. (1984). *Classification and Regression Trees.* Chapman and Hall, 1st edition edition.

Buhlmann, P. and Yu, B. (2002). Analyzing bagging. *Annals of Statistics*, 30(4):927–961.

Coppersmith, D., Hong, S., and Hosking, J. (1999). Partitioning nominal attributes in decision trees. *Data Mining and Knowledge Discovery*, 3:197–217.

De'ath, G. mvpart: Multivariate partitioning, 1.3-1. *R Package Version*.

Diaz-Uriarte, R. and de AndrŠs, S. A. (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7:3–15.

Efron, B. (2004). The estimation of prediction error: Covariance penalties and cross-validation. *Journal of the American Statistical Association*, 99(467):619–642.

Efron, B. and Tibshirani, R. (1997). Improvements on cross-validation: The .632+ bootstrap method. *Journal of the American Statistical Association*, 92(438):548–555.

Freund, Y. and Schapire, R. (1996). Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156.

Friedman, J. and Hall, P. (2000). On bagging and nonlinear estimation. *Journal of Statistical Planning and Inference*, 137(3):669–683.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63:3–42.

Goldstein, B., Polley, E., and Briggs, F. (2011). Random forests for genetic association studies. *Statistical Applications in Genetics and Molecular Biology*, 10(1):1–34.

Hammann, F., Gutmann, H., Vogt, N., Helma, C., and Drewe, J. (2010). Prediction of adverse drug reactions using decision tree modeling. *Clinical Pharmacology and Therapeutics*, 88:52–59.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2nd edition edition.

Ho, T. (1995). Random decision forest. *Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC*, 14-16:278–282.

Ho, T. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844.

Ishwaran, H., Kogalur, U., Blackstone, E., and Lauer, M. (2008). Random survival forests. *The Annals of Applied Statistics*, 2(3):841–860.

Jiang, P., Wu, H., Wang, W., Ma, W., Sun, X., and Lu, Z. (2007). Mipred: classification of real and pseudo microrna precursors using random forest prediction model with combined features. *Nucleic Acids Research*, 35(2):339–344.

Kumar, M. and Thenmozhi, M. (2006). Forecasting stock index movement: A comparison of support vector machines and random forest. *Indian Institute of Capital Markets 9th Capital Markets Conference.*

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

Lin, Y. and Jeon, Y. (2006). Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590.

Meinshausen, N. (2006). Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999.

Olshen, R. (2010). Remembering leo breiman. *The Annals of Applied Statistics*, 4(4):1644–1648.

Pal, M. (2003). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222.

Palmer, D., O'Boyle, N., Glen, R., and Mitchell, J. (2007). Random forest models to predict aqueous solubility. *J Chem Inf Model*, 47(1):150–158.

Segal, M. and Xiao, Y. (2011). Multivariate random forests. *WIREs Data Mining and Knowledge Discovery*, 1:80–87.

Shi, T., Seligson, D., Belldegrun, A., Palotie, A., and Horvath, S. (2005). Tumor classification by tissue microarray profiling: Random forest clustering applied to renal cell carcinoma. *Modern Pathology*, 18(4):547–557.

Siroky, D. (2009). Navigating random forests and related advances in algorithmic modeling. *Statistics Surveys*, 3:147–163.

Ward, M., Pajevic, S., Dreyfuss, J., and Malley, J. (2006). Short-term prediction of mortality in patients with systemic lupus erythematosus: Classification of outcomes using random forests. *Arthritis and Rheumatism*, 55:74–80.

CHAPTER 3. ITERATIVE BIAS CORRECTION IN RANDOM FORESTS

Abstract

Random forests (RFs) may produce biased predictions in regression problems. The default bias correction method implemented in the R package *randomForest* often does not work well. Breiman suggested an iterated bagging procedure to decrease both variance and bias of predictions in general, but this bias correction procedure is seemingly not well-known nor well-studied with regard to predictions by RF (Breiman 2001a). We examine this iterative bagging idea in RFs as an alternative bias correction to the common R default. This is done by generating a second RF based on out-of-bag prediction errors in order to estimate the bias for an independent test case. Real data examples show that this new bias correction dramatically reduced prediction errors compared to both standard RFs without bias corrections and RFs with the default bias correction method implemented in the *randomForest* package. We found that the iterative RF method worked well in some real datasets and can be tuned by cross validation.

1 Introduction

Random forest (RF) methodology is a useful data mining technique for both regression and classification problems (Breiman 2001a,b). A RF is a collection of classification or regression trees generated by a bootstrap procedure. In each tree, the RF algorithm separates the regressor space into disjoint hyperrectangles and observations in each hyperrectangle are aggregated to estimate or predict the response surface. Lin and Jeon (2006) showed that a RF can be viewed as an adaptively weighted $k-$nearest neighbor predictor. By averaging all tree predictions, a RF produces a weighted average of the responses of training cases to predict a test case, where large weights are assigned to the training cases in close proximity to the test case.

In regression problems, linear models also predict the response of an independent test case by a weighted average (linear combination) of all training cases. While linear models can assign negative weights to some training cases, RFs assign only nonnegative weights that must sum to one. This property of RFs may result in prediction biases, as can be illustrated with the following simple simulation example.

Suppose a dataset has two predictor variables X_1 and X_2 and a response variable Y. We hope to grow a RF based on a training set, and use this RF to predict an independent test case based on its covariates (X_1, X_2). Suppose we have a training set $\boldsymbol{\mathcal{C}}$ of size n, where X_1 equally partitions the interval [0.1, 0.9],

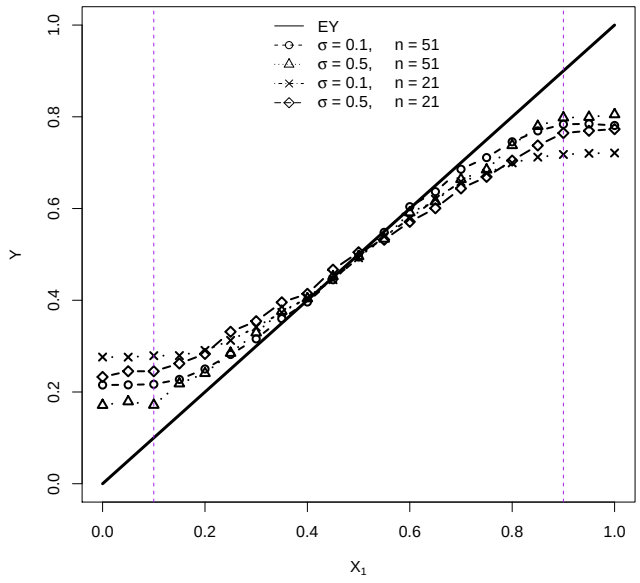


Figure 3: Average predictions by RFs, indicating bias (deviation) from the true reponse mean (solid line). These results are based on prediction averaged over 1000 simulations.

$X_2 \sim U(0, 1)$, and $Y|X_1, X_2 \sim N(X_1, \sigma^2)$. A RF containing 100 trees is grown based on $\boldsymbol{\mathcal{C}}$.

To easily visualize the prediction performance by the RF, a test set $\boldsymbol{\mathcal{H}}$ is independently generated such that there are 21 test cases with X_1values equally partitions the unit interval [0, 1], and X_2 values randomly generated from $U(0, 1)$. The response value Y is simulated from the conditional distribution of $Y|X_1, X_2$. All parameters for building RFs are the same as the default settings in the R package *randomForest* (Liaw and Wiener 2002). This process is repeated 1000 times with different σ and n combinations. The average of predictions for test cases are plotted in Figure 3 with different X_1 values.

From Figure 3 we can see that predictions by RFs can be biased, especially when n is small and when extrapolation is needed (on the two edges of the unit interval). This is intuitively easy to understand. RF method is fundamentally a nearest neighbor method. When there are not many training cases available near a test case, the RF has to predict the test case using training case responses whose mean response may differ substantially from the mean response of the target test case. For a test case with an extreme X_1 value, the RF algorithm tends to use its nearest neighbors (in terms of X_1 distance in this example) to predict it. The unbalanced nature of the data points on the edges tends to predict this particular case "regressively" towards the center of the training data.

Bagging (Breiman 1996) is a general procedure for improving predictions, which has been shown effective in reducing prediction variance but having little impact on reducing prediction bias. Breiman

(2001a) developed an iterative bagging algorithm to reduce both variance and bias in general prediction problems. Recently, Zhang and Lu (2012) proposed a simple noniterative version of this RF bias correction idea and found that it compares favorably with other bias correction approaches. In the current paper, we point out the connection between Breiman 2001a and Zhang and Lu 2012, investigate the value of iteration, and compare strategies for identifying the optimal number of iterations for correcting the bias of RF. The iterative bagging approach of (Breiman 2001a) is seemingly not well-known or understood in application to improving RF predictions. We provide evidence that this approach performs better for debiasing and improving RF predictions than a standard debiasing technique which is currently the default of the R package *randomForest* (see Section 2).

In Section 2, we present the details of a single-iteration bias correction method in RFs for regression problems. In Section 3, we test its performance using some real data examples and find that this bias correction method dramatically improves predictions by RFs across many test datasets. In Section 4, we generalize the bias correction idea in RFs by applying the bias correction idea iteratively. We also propose using cross-validation to select the number of iterations.

2 Bias Correction for Random Forests

From Figure 3 we know that the predictions given by RFs can have biases, and these biases can vary from data point to data point. In the R package *randomForest*, there is a simple bias correction method, as described BC_1 below.

1) Suppose we have a training set of size n, $\mathcal{C} = \{C_1, \ldots, C_n\}$, where a training case C_i has predictor variables $\boldsymbol{X}_i$ and a continuous response variable Y_i. To grow individual trees, bootstrap resampling is done on $\mathcal{C}$. A large number of bootstrap datasets are created and a tree is grown on each, where each dataset is constructed by independently and with replacement sampling n cases from $\boldsymbol{\mathcal{C}}$. We denote $\mathcal{T}(C_i)$ as the set of trees that do not contain training case C_i.

BC_1: Bias Correction of RFs in R Package *randomForest*

2) We denote $T_j(\boldsymbol{X}_i)$ to be the predicted response value for case C_i by tree T_j. Then the out-of-bag prediction of C_i by this RF is

$$\tilde{Y}_i = \frac{\sum_{\{j \in \mathcal{T}(C_i)\}} T_j(\boldsymbol{X}_i)}{||\mathcal{T}(C_i)||}, \tag{10}$$

where $||\mathcal{T}(C_i)||$ is the number of elements in set $\mathcal{T}(C_i)$.

3) Let $\hat{\alpha}$ and $\hat{\beta}$ denote the estimated intercept and slope, respectively, obtained by fitting a simple linear regression of the true training case response Y_i on its out-of-bag prediction $\tilde{Y}_i$, $i = 1, \ldots, n$.

4) For a test case C_0 with predictor variable $\boldsymbol{X}_0$, let $\tilde{Y}_0$ be the direct predicted response from the forest. Then the bias-corrected prediction is

$$\hat{Y}_0 \quad = \hat{\alpha} + \hat{\beta}\tilde{Y}_0 \quad . \tag{11}$$

BC_1 says that the default bias correction method in R package *randomForest* assumes a linear relationship between the true response values and the out-of-bag predicted response values. It should be clear that this algorithm simply assumes a linear relationship between the bias corrected prediction value $\hat{Y}$ and the naive (uncorrected) prediction value $\tilde{Y}$, without using any regressor information in the final step to correct bias. We will show that this bias correction method does not work well in some situations because bias of RF predictions can vary non-linearly across different regressor regions.

We can see this from the following simple example. Suppose (X, Y) has a joint distribution shown below

$$\begin{aligned} X &\sim U(0, 1) \\ Y &= \begin{cases} X + \epsilon & \text{if } X \leq 0.5 \\ X - 0.5 + \epsilon & \text{if } X > 0.5, \end{cases} \end{aligned} \tag{12}$$

where $\epsilon \sim N(0, 0.1^2)$ is independent of X. RFs were grown based on 50 training cases generated from (12), and we hope to predict the response value of independent test cases based on their X values. The results are displayed in Figure 4. We see in Figure 4 that the prediction at $X = 0.5$ is more negatively biased than that at $X = 1.0$, although their expected response values are equal. This is because RF is fundamentally a k-nearest neighbor method (Lin and Jeon 2006). RF prediction of a test case is simply the average over training-case response values in a neighborhood of the test case. The training cases with predictor values a little larger than 0.5 negatively affect predictions for test cases with X values close to but less than 0.5. Likewise, predictions for test cases slightly greater than 0.5 are positively biased by nearby training cases with X values slightly less than 0.5. Figure 4 shows that the predictions after bias correction by BC_1 are even worse than the uncorrected RF predictions. To address this problem, we examine a different bias correction idea for RF predictions, as shown in BC_2.

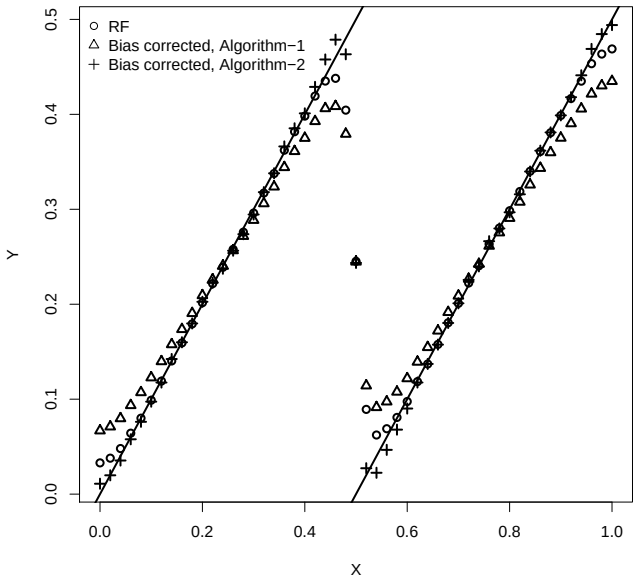


Figure 4: Comparison of RF predictions based on different bias corrections . The diagonal lines indicate the mean response function. The symbols show average RF predictions over 1000 simulations for various values of X, on uncorrected RF predictions ($\circ$), BC_1 RF predictions (Δ) and BC_2 RF predictions ($+$).

BC_2: Bias Correction of RFs Using a Second RF

1) Using the same notation as in BC_1, we grow a RF, namely RF_p (p for prediction), based on a training set $\boldsymbol{\mathcal{C}}$ of size n, with response Y_i and vector of predictors $\boldsymbol{X}_i$, $i = 1, \ldots, n$.

2) The out-of-bag prediction for training case $\mathcal{C}_i$, $\tilde{Y}_i$ ($i = 1, \ldots, n$) is obtained from $\mathcal{T}(C_i)$. Then the out-of-bag estimation of bias for C_i is defined as

$$\tilde{B}_i \quad = \quad \tilde{Y}_i - Y_i. \tag{13}$$

3) Grow a second RF, namely RF_b (b for bias estimation), with response $\tilde{B}_i$'s and vector of predictors $\boldsymbol{X}_i$ ($i = 1, \ldots, n$).

4) For a test case C_0 with predictor vector $\boldsymbol{X}_0$, let $\tilde{Y}_0$ be the predicted response from RF_p, and $\tilde{B}_0$ be the estimated bias from RF_b. Then the bias corrected prediction for C_0 is

$$\hat{Y}_0 \quad = \quad \tilde{Y}_0 - \tilde{B}_0. \tag{14}$$

Table 4: Description of real datasets used in this paper.

Dataset	Name	No. of predictors	No. of observations
1	Concrete Compressive Strength	8	1030
2	Housing	13	506
3	Computer Hardware	7	209
4	Auto MPG	7	398
5	Auto Price	19	201
6	Servo	4	167
7	Forest Fire	12	517

In Figure 4 we see that our bias correction method (BC_2) worked much better than both the standard RF and the bias correction method implemented in R package *randomForest* (BC_1). Standard RF prediction have severe bias on cases near edges or change points in the predictor domain. However BC_2 almost eliminated the bias for all values of X in this simulation example. This algorithm is an implementation of Breiman's debiasing idea (Breiman 2001a) in decision trees, which was mentioned in (Zhang and Lu 2012).

3 Real Data Examples

We applied our bias correction method to a few real datasets available on the UCI data repository. There are 23 datasets with numerical response values on this website. We chose the seven datasets with 100-2000 observations in our study. Some basic information about the datasets is provided in Table 4. To measure the prediction performance of a RF method on each dataset, we carried out the following procedure for each combination of method (standard RF, bias-corrected RF by BC_1 and BC_2) and dataset. A dataset was randomly partitioned into a training set containing 2/3 data and a test set containing the remaining 1/3 data. A RF was generated from the cases in the training set and used to predict the responses for cases in the test set. This process was repeated independently $M = 1000$ times. Let $\boldsymbol{\mathcal{C}}_m$ and $\boldsymbol{\mathcal{H}}_m$ denote the mth training and test sets, respectively. For $m = 1, \ldots, M$, $\hat{f}_{\boldsymbol{\mathcal{C}}_m}(\boldsymbol{X}_i)$ denote the RF prediction corresponding to training set $\boldsymbol{\mathcal{C}}_m$ and test case $C_i = (\boldsymbol{X}_i, Y_i) \in \boldsymbol{\mathcal{H}}_m$. We define the estimated prediction mean squared error by

$$\widehat{MSE} = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{||\boldsymbol{\mathcal{H}}_m||}\sum_{C_i\in\boldsymbol{\mathcal{H}}_m}[\hat{f}_{\boldsymbol{\mathcal{H}}_m}(\boldsymbol{X}_i) - Y_i]^2. \quad (15)$$

This hold-out estimator is similar to a cross-validation estimator, and has been widely used as an estimator for the quantity $E_{\boldsymbol{\mathcal{C}}}E_{\boldsymbol{\mathcal{H}}}[\hat{f}_{\boldsymbol{\mathcal{C}}}(\boldsymbol{X}) - Y]^2$ (Borra and Di Ciaccio 2010, Hastie et al. 2009).

Table 5: Comparison of prediction performance for three RF methods based on the datasets shown in Table 4. The prediction errors for standard RF (column 2), RF bias corrections by BC_1 (column 3), BC_2 (column 4), and BC_3 tuned by 10-fold cross-validation (column 5) are presented in this table. The last column shows the MSE by the best method for comparative purposes only, where the best MSE is determined from Figure 3 (i.e., based on the best number of iterations for use in BC_3 noting that standard RF and BC_2 are implementations of BC_3 with a fixed number of iterations.

Dataset	Prediction MSE in (15)				
	Standard RF	RF (BC_1)	RF (BC_2)	Tuned BC_3	Best method
1	34.43	28.83	20.99	21.46	20.99
2	11.96	11.09	10.35	11.02	10.35
3	2911	2163	1262	1365	1262
4	8.61	8.39	7.72	8.35	7.72
5	5.52×10^6	5.16×10^6	5.13×10^6	5.25×10^6	5.13×10^6
6	0.756	0.542	0.446	0.364	0.338
7	4106	3841	4603	4013	3841

The estimated prediction MSEs for the three methods are presented in Table 5. Table 5 shows that BC_1 reduced prediction error in all datasets compared to standard RFs. Bias correction by BC_2 further improved RF predictions substantially except in dataset *7*. For most of the other datasets, the reduction in $\widehat{MSE}$ obtained by BC_2 relative to standard random forests was more than twice the reduction obtained from the R default bias correction provided by BC_1.

4 Generalization of Bias Correction in Random Forests

In Section 2, we described the idea of correcting bias when predicting Y in RFs by growing a second RF (RF_b) using the out-of-bag bias estimates as the response. We can apply a similar bias correction when predicting the out-of-bag bias esimates from RF_b. Furthermore, such a process could be repeated iteratively. This iterative procedure has been suggested by Breiman as a general method of debiasing for bagging (Breiman 2001a). We describe this bias correction method of tree implementation in BC_3.

BC_3: Iterative Bias Correction in RFs

1) Given a training set $\boldsymbol{\mathcal{C}} = \{(\boldsymbol{X}_i, Y_i),\ i = 1, \ldots, n\}$, let $b_i^1 = Y_i$, for $i = 1, \ldots, n$.

2) At the kth iteration, $k = 1, \ldots, K$,

grow a RF (called RF^k) using the training data $\{(\boldsymbol{X}_i, b_i^k) : i = 1, \ldots, n\}$.

for each training case $C_i^k = (\boldsymbol{X}_i, b_i^k)$, estimate its out-of-bag prediction $\tilde{b}_i^k$ using $\mathcal{T}^k(C_i^k)$, the trees in RF^k that do not contain C_i^k.

For all $i = 1, \ldots, n$, set $b_i^{k+1} = b_i^k - \tilde{b}_i^k$.

3) To predict the response for a new test case C_0 with predictor vector $\boldsymbol{X}_0$, sum the predicted values

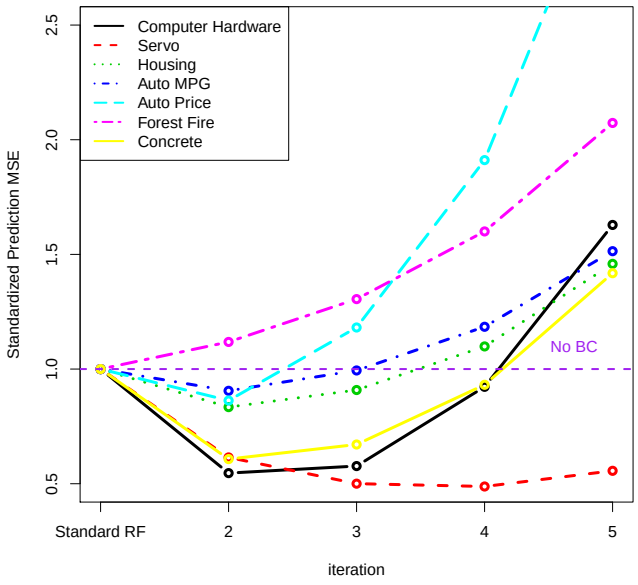


Figure 5: Performance of iterative bias corrections on the datasets indicated in Table 4. These results were obtained by averaging the prediction MSEs over 1000 simulations. Then for each dataset, the prediction MSE values were standardized by dividing by the MSEs by standard RFs without bias correction. So the horizontal line $y = 1$ correspond to the performance by standard RFs for all seven datasets. In the BC_3 results above, note standard RF corresponds to $NI = 1$ while $NI = 2$ corresponds to the bias correction procedure described in BC_2.

from all RFs grown in step 2), i.e., use

$$\hat{Y}_0 \quad = \quad \sum_{k=1}^{K} \tilde{b}_i^k, \tag{16}$$

where $\tilde{b}_i^k$ is the prediction obtained from RF^k at $\boldsymbol{X}_0$. $NI = 1$ corresponds to standard (uncorrected) RF and $NI = 2$ corresponds to BC_2.

In order to test the performance of iterative bias corrections in RFs (BC_3), we conducted similar analyses on the seven real datasets indicated in Table 4. The prediction MSEs were obtained based on 1000 random partitions of each dataset as in Table 5. From Figure 5, we see that all datasets except "Forest Fire" were improved by one iteration of bias correction (BC_2), which is exactly the Table 5 results discussed in Section 3. Additional iterations of bias corrections further reduced prediction errors only in the "Servo" dataset, where the optimal performance was achieved at 4 iterations.

It should be clear that the number of iterations (NI) in BC_3 is a complexity parameter for this bias correction method. A large NI may result in overfitting problems. The optimal NI value varies from

dataset to dataset (see Figure 5). Breiman investigated the number of iterations with an empirical way (Breiman 2001a). He suggested to stop iterative debiasing when the mean squared errors for new cases from the next iteration are q times of the minimal errors so far, where he took q to be 1.2, 1.1 or 1.05. This "tuning" method is rather arbitrary in selecting the value q.

We suggest determining this tuning parameter using cross validation. For each of the real datasets listed in Table 4, we randomly split the dataset to two parts, a training set (about 2/3 of the cases) and a test set (the remaining), and performed a 10-fold cross validation on the training set to select the optimal bias correction method for RFs. The best method was selected based on the mean squared prediction errors on the training set by cross validation. This procedure was repeated 1000 times, and the mean squared errors by this tuned BC_3 method are presented in column 5 of Table 5. The tuning by cross-validation gave good predictions that are close to the best method (last column).

5 Conclusion

In this paper we independently proposed a bias correction idea in random forests. This is an application to RFs of a general bias (and prediction) correction strategy of (Breiman 2001a), which has seemingly not received much consideration for improving RF predictions. Simulation and real data examples show that this correction method dramatically improves predictions in RFs over both the standard RFs and the RFs implemented with the default bias correction in R package *randomForest*. Motivated by this idea, we generalized the bias correction method by conducting it with more iterations. The real data example (Servo) indicated that predictions on some datasets may be improved by more iterations of bias correction, through often one iteration of bias correction performs quite well. It is feasible to tune this complexity parameter by cross validation, which often produces bias-corrected RF predictions that are similar to using the best number of iterative bias corrections.

References

Amaratunga, D., Cabrera, J., and Lee, Y.-S. (2008). Enriched random forests. *Bioinformatics*, 24(18):2010–2014.

Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588.

Borra, S. and Ciaccio, A. D. (2010). Measuring the prediction error. a comparison of cross-validation, bootstrap and covariance penalty methods. *Computational Statistics and Data Analysis*, 54:2976–2989.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, 26(2):123–140.

Breiman, L. (1996b). Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, 24(6):2350–2383.

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

Breiman, L. (2001b). Statistical modeling: The two cultures (with rejoinder). *Statistical Science*, 16(3):199–231.

Breiman, L., Friedman, J., Stone, C., and Olshen, R. (1984). *Classification and Regression Trees.* Chapman and Hall, 1st edition edition.

Buhlmann, P. and Yu, B. (2002). Analyzing bagging. *Annals of Statistics*, 30(4):927–961.

Coppersmith, D., Hong, S., and Hosking, J. (1999). Partitioning nominal attributes in decision trees. *Data Mining and Knowledge Discovery*, 3:197–217.

De'ath, G. mvpart: Multivariate partitioning, 1.3-1. *R Package Version*.

Diaz-Uriarte, R. and de AndrŠs, S. A. (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7:3–15.

Efron, B. (2004). The estimation of prediction error: Covariance penalties and cross-validation. *Journal of the American Statistical Association*, 99(467):619–642.

Efron, B. and Tibshirani, R. (1997). Improvements on cross-validation: The .632+ bootstrap method. *Journal of the American Statistical Association*, 92(438):548–555.

Freund, Y. and Schapire, R. (1996). Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156.

Friedman, J. and Hall, P. (2000). On bagging and nonlinear estimation. *Journal of Statistical Planning and Inference*, 137(3):669–683.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63:3–42.

Goldstein, B., Polley, E., and Briggs, F. (2011). Random forests for genetic association studies. *Statistical Applications in Genetics and Molecular Biology*, 10(1):1–34.

Hammann, F., Gutmann, H., Vogt, N., Helma, C., and Drewe, J. (2010). Prediction of adverse drug reactions using decision tree modeling. *Clinical Pharmacology and Therapeutics*, 88:52–59.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2nd edition edition.

Ho, T. (1995). Random decision forest. *Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC*, 14-16:278–282.

Ho, T. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844.

Ishwaran, H., Kogalur, U., Blackstone, E., and Lauer, M. (2008). Random survival forests. *The Annals of Applied Statistics*, 2(3):841–860.

Jiang, P., Wu, H., Wang, W., Ma, W., Sun, X., and Lu, Z. (2007). Mipred: classification of real and pseudo microrna precursors using random forest prediction model with combined features. *Nucleic Acids Research*, 35(2):339–344.

Kumar, M. and Thenmozhi, M. (2006). Forecasting stock index movement: A comparison of support vector machines and random forest. *Indian Institute of Capital Markets 9th Capital Markets Conference.*

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

Lin, Y. and Jeon, Y. (2006). Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590.

Meinshausen, N. (2006). Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999.

Olshen, R. (2010). Remembering leo breiman. *The Annals of Applied Statistics*, 4(4):1644–1648.

Pal, M. (2003). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222.

Palmer, D., O'Boyle, N., Glen, R., and Mitchell, J. (2007). Random forest models to predict aqueous solubility. *J Chem Inf Model*, 47(1):150–158.

Segal, M. and Xiao, Y. (2011). Multivariate random forests. *WIREs Data Mining and Knowledge Discovery*, 1:80–87.

Shi, T., Seligson, D., Belldegrun, A., Palotie, A., and Horvath, S. (2005). Tumor classification by tissue microarray profiling: Random forest clustering applied to renal cell carcinoma. *Modern Pathology*, 18(4):547–557.

Siroky, D. (2009). Navigating random forests and related advances in algorithmic modeling. *Statistics Surveys*, 3:147–163.

Ward, M., Pajevic, S., Dreyfuss, J., and Malley, J. (2006). Short-term prediction of mortality in patients with systemic lupus erythematosus: Classification of outcomes using random forests. *Arthritis and Rheumatism*, 55:74–80.

CHAPTER 4. CASE-SPECIFIC RANDOM FORESTS

Ruo Xu, Dan Nettleton and Daniel J. Nordman

Ruo Xu is Graduate Student, Department of Statistics, Iowa State University, Ames IA 50011 (Email: xuruo@iastate.edu); Dan Nettleton is Professor, Department of Statistics, Iowa State University, Ames, IA 50011 (Email: dnett@iastate.edu); Daniel J. Nordman is Associate Professor, Department of Statistics, Iowa State University, Ames, IA 50011 (Email: dnordman@iastate.edu). This work was supported by National Science Foundation-Plant Genome Award 0922746.

Abstract

Random forest (RF) methodology is a nonparametric methodology for prediction problems. A standard way to utilize RFs includes generating a global RF in order to predict all test cases of interest. In this paper, we propose growing different RFs specific to different test cases, namely case-specific random forests (CSRFs). In contrast to the bagging procedure in the building of standard RFs, the CSRF algorithm takes weighted bootstrap resamples to create individual trees, where we assign large weights to the training cases in close proximity to the test case of interest a priori. Tuning methods are discussed to avoid overfitting issues. Both simulation and real data examples show that CSRFs have better performance than standard RFs in prediction. We also propose the idea of case-specific variable importance (CSVI), a way to compare the relative predictor variable importance for predicting a particular case. It is possible that the idea of building a predictor case-specifically can be generalized in other areas.

Keywords: Prediction; Variable Importance; Machine Learning.

1 Introduction

Random forest (RF) methodology is a useful machine learning technique for regression and classification problems (Breiman 2001a). The popularity of random forests (RFs) is reflected by its extension and incorporation in other methodology, such as multivariate random forests (De'ath; Segal and Xiao 2011) , quantile regression forests (Meinshausen 2006), enriched random forests for microarray analysis (Amaratunga et al. 2008), random survival forests (Ishwaran et al. 2008) and the R package "*pathwayRF*" for metabolic pathway analysis (Pang et al 2006), etc. The RF approach has also been found to work well in high dimensional problems (Breiman 2001b).

The standard implementation of RF creates a collection of classification or regression trees through bootstrap resampling, which can then be combined for prediction. To motivate what follows, we briefly recall the mechanics in a prediction problem with a standard data-generating model given by

$$Y = f(\boldsymbol{X}) + \epsilon,$$

where $\boldsymbol{X}$ is a p-dimensional predictor variable, Y is a response random variable, and ϵ is a mean-zero error term. Based on a training dataset $\boldsymbol{\mathcal{C}} = \{C_i,\, i = 1, \ldots, N\}$ of N cases $C_i = (\boldsymbol{X}_i,\, Y_i)$ of paired regressor/response values, the standard version of RF independently and uniformly resamples cases from $\boldsymbol{\mathcal{C}}$ to create a bootstrap dataset $\boldsymbol{\mathcal{C}}^* = \{C_i^*,\, i = 1, \ldots, N\}$ from which a regression tree T^* is grown. Repeating this process B times produces a series of bootstrap datasets $\boldsymbol{\mathcal{C}}_j^*$ and associated trees T_j^*, $j = 1, \ldots, B$, that constitute a "random forest". To obtain a prediction $\hat{Y}(\boldsymbol{X}_0)$ for the response at a given regressor value $\boldsymbol{X}_0$, each tree T_j^* yields a prediction $\hat{Y}_j(\boldsymbol{X}_0)$ at $\boldsymbol{X}_0$, and these predictions are averaged across all trees to get the final RF prediction $\hat{Y}(\boldsymbol{X}_0) = \sum_{j=1}^{B} \hat{Y}_j(\boldsymbol{X}_0)/B$. Note that, to obtain a prediction at a different regressor value $\tilde{\boldsymbol{X}}_0$, the process is simply repeated using the *same* trees.

The standard RF implementation intrinsically builds trees without any regard to where predictions are desired. In this sense, the standard RF method essentially treats predictions at any regressor $\boldsymbol{X}_0$ as equally important. However, in many cases, a *reverse* approach may be beneficial. That is, in some situations one might start by specifying a point $\boldsymbol{X}_0$ of interest in the regressor space and then consider building trees which are tailored for prediction at $\boldsymbol{X}_0$. As an example to be illustrated more fully later, a home seller may be far more interested in predicting a sales price based on the building characteristics of her/his house rather than predicting prices for houses in general. Our motivation here is to develop such a RF methodology that is specific to any given case of interest, as opposed to using one collection of trees for predicting any case in the standard RF method. Intuitively, given a prediction point $\boldsymbol{X}_0$ of interest,

we wish to grow a RF by focusing on the cases in the data which are "close" to or resemble $\boldsymbol{X}_0$. To this end, we propose a scheme for creating RF trees especially customized for prediction of Y at $\boldsymbol{X}_0$ through a proximity-based weighted resampling. We call this method a *Case-Specific Random Forest* (CSRF). For several data models and real datasets, we show that the CSRF almost always outperforms the standard version of RF in terms of prediction MSE.

The rest of the manuscript is organized as follows. In Section 2 we introduce the algorithm for building CSRFs by weighted bootstrap resamples. Section 3 then presents some simulation results and real data examples to examine the prediction performance of CSRFs in comparison to standard RFs. In Section 4 we discuss a device for assessing the importance of predictor variables with CSRFs. This approach necessarily differs from the variable importance measure in standard RF methodology, which is not appropriate in the case-specific prediction setting. To illustrate the proposed methodology and demonstrate its relevance in practice, Section 5 provides a real data example involving the sales price prediction for a house in Ames, Iowa. An Appendix describes some alternative implementations of the new CSRF method, where the resulting predictions are also better than the standard RF.

2 Case-Specific Random Forests by Weighted Bootstrap

Rather than using one set of trees for all predictions in the standard RF approach, we describe here how different RFs may be built specifically for a given regressor setting $\boldsymbol{X}_0$ at which a prediction is desired.

When a RF is constructed, instead of creating trees from bootstrap samples based on uniform resampling from the dataset cases, we apply a weighted resampling scheme that assigns higher probability weights to those data cases in close proximity of $\boldsymbol{X}_0$. The distance between observations is complicated in high dimensional problems, and "proximity" is a complex combination of both the response variable and the relevant predictor variables. The standard RF methodology provides a proximity measure for the purpose of assigning distances between observations (Breiman (2001a)). This proximity measure is defined as follows. Given a training dataset $\boldsymbol{\mathcal{C}} = \{C_1, \ldots, C_N\}$ and a desired regressor location $\boldsymbol{X}_0$ for prediction, B trees are grown by resampling from $\mathcal{R}$, as described in Section 1. For case $C_i = (\boldsymbol{X}_i, Y_i)$, let E_i denote the number of these trees containing both $\boldsymbol{X}_0$ and $\boldsymbol{X}_i$ in the same terminal node. Then an "proximity weight" for training case C_i relative to $\boldsymbol{X}_0$ can be defined as

$$D_i \quad = \quad \frac{E_i}{\sum_{j=1}^{N} E_j}, \quad i = 1, \ldots, N. \tag{17}$$

The weights $D_1, \ldots, D_N$ define a probability distribution on cases in the training data $\boldsymbol{\mathcal{C}}$, which can be

used for building trees with observations close to $\boldsymbol{X}_0$ based on weighted bootstrap resamples from $\boldsymbol{\mathcal{C}}$.

For a training set $\mathcal{R}$ and prediction point $\boldsymbol{X}_0$, we now give the algorithm for building CSRFs.

CSRF Algorithm

1) A standard RF, say RF^w(w for "weight defining"), is grown based on $\boldsymbol{\mathcal{C}} = \{C_1, \ldots, C_N\}$, where each tree has the same maximal terminal node (MTN) size denoted as MTN^w.

2) From RF^w and $\boldsymbol{X}_0$, proximity weights on cases $\boldsymbol{\mathcal{C}}$ are defined as $(D_1, \ldots, D_N)$ from (1).

3) A second RF, RF^p(p for prediction), is grown where each bootstrap resample $R_j^* = \{C_1^*, \ldots, C_N^*\}$ is obtained by sampling with replacement from $\boldsymbol{\mathcal{C}}$ using the probabilities $(D_1, \ldots, D_N)$; this produces corresponding trees $T_1^*, \ldots, T_{B_p}^*$ that define RF^p, where B_p is the number of bootstrap resamples for defining RF^p. In growing these trees, the standard default maximal node size is used (e.g., 5 in regression, Liaw and Wiener 2002).

4) Submit $\boldsymbol{X}_0$ to RF^p and get the prediction $\hat{Y}(\boldsymbol{X}_0)$, as the usual sample average of the tree predictions.

Note that the number of trees grown in RF^w and RF^p, denoted above as B_w and B_p respectively, need not be the same. In our numerical studies, we used 10 times more trees in RF^w than in RF^p to reduce the uncertainty in the importance weights $D_1, \ldots, D_N$ for the weighted bootstrap.

From the definition of the proximity measure (17) and the CSRF algorithm, we see that the maximal terminal nodesize MTN^w for RF^w determines how widely the weights $(D_1, \ldots, D_N)$ are placed on training cases $\boldsymbol{\mathcal{C}} = \{C_1, \ldots, C_N\}$ for building RF^p via a weighted bootstrap. Larger MTN^w values tend to produce nonzero weights on more cases among $\boldsymbol{\mathcal{C}}$. At the extreme, a MTN^w value larger than the size N of training set (denoted as $MTN^w = \infty$ here) will generate uniform weights $D_i = 1/N$, in which case the corresponding RF^p simply amounts to a standard RF. Hence, MTN^w is a tuning parameter for the CSRF procedure. However, as will be illustrated in Section 3, the CSRF across several fixed choices of MTN^w typically outperformed the standard RF in our simulations no matter which of several choices of MTN^w (5, 10, 20, 30) were used for CSRF construction.

We may illustrate the mechanics of the CSRF method in more detail using a simple example. Suppose we have a training set containing two cases $\boldsymbol{\mathcal{C}} = \{C_1, C_2\}$, $C_1 \equiv (X = 0, Y_1)$ and $C_2 \equiv (X = 1, Y_2)$. The relationship between the response variable Y and the only predictor variable X is given by $Y|X \sim N(f(X), \sigma^2)$; that is, Y is conditionally normally distributed with a trend function $f(X)$ where $f(0) \neq f(1)$ here. Suppose we have a prediction point $X_0 = 0$ of interest and wish to predict its corresponding

response by both standard RF and CSRF. The standard RF by the usual resampling scheme generates bootstrap datasets (C_1, C_1), (C_1, C_2) and ($C_{2,}$, C_2) with probabilities proportional to 1:2:1, so that the final prediction for $X_0 = 0$ follows as $\frac{3}{4}Y_1 + \frac{1}{4}Y_2$. The proximity-based importance weights given by RF^w are $D_1 = \frac{3}{4}$ and $D_2 = \frac{1}{4}$ for fully grown trees having $MTN^w = 1$. Therefore, weighted resamping generates bootstrap samples (C_1, C_1), (C_1, C_2) and (C_2, C_2) with probabilities proportional to 9:6:1, and hence the final prediction for $X_0 = 0$ follows as $\frac{15}{16}Y_1 + \frac{1}{16}Y_2$. From this example, we see that the resamples by CSRF more heavily select the training cases (C_1 in this example) close to the point X_0 of prediction compared to the standard RF. Consequently, CSRF provides a type of higher resolution for prediction by focusing on the nearby observations, which can play an important role in reducing prediction bias. For comparison, the prediction MSEs in the above example are

$$\left(1 + \frac{113}{128}\right)\sigma^2 + \frac{[f(0) - f(1)]^2}{256} \quad \text{and} \quad \left(1 + \frac{80}{128}\right)\sigma^2 + \frac{[f(0) - f(1)]^2}{16}$$

for CSRF and standard RF, respectively. One observes that the bias contribution, particularly for large $[f(0) - f(1)]^2$, can be significantly reduced at the price of small increase in the proportionality constant for error variance σ^2.

3 Simulation and Real Data Examples

We examined the prediction performance of CSRF, considering both simulation from several data-generating models (with nine data models in Table 6) as well as randomized cross-validation over real data examples (seven datasets listed in Table 7). For comparison, prediction MSEs from both standard RFs and CSRFs were obtained, and both methods were implemented in similar fashions with final predictions based on 100 trees/forest and the default MTN of 5 for the standard RF and for RF^p in CSRF. The remaining tuning parameter MTN^w in the CSRF method was fixed and evaluated at 5, 10, 20 or 30 for all numerical examples. See Tables 6 and 7 for the details on defining the prediction MSEs in the simulation and cross-validation studies, respectively.

Table 8 lists the average prediction MSEs with CSRFs and standard RFs for each data model/set in Tables 6 and 7. As shown in Table 8, CSRFs outperformed standard RFs, *regardless* of MTN^w values, over 13 simulation examples. In fact, in all 17 data models/sets considered, CSRFs performed comparably or better than standard RFs over all fixed MTN^w settings, and there always existed several choices of MTN^w for which the CSRF method was strictly better than standard RF in each simulation example.

Table 6: Description of the data-generating models used in simulation evaluation of prediction performance. Each data model uses 10 IID predictor variables from either uniform $U(0,\,1)$, chi-square $\chi^2(1)$ or $Exp(1)$ (exponential mean 1) distributions and independent random errors ϵ from $N(0,\,0.1^2)$; only some of the predictor variables are functionally related to the response as indicated above. For each model, we conducted 1000 simulation runs, where in each run 50 training cases and 100 test cases were randomly and independently generated from the joint distribution of $(\boldsymbol{X},\,Y)$. The training set was used to implement both standard RFs and CSRFs, and prediction errors were then calculated and averaged over the test set. Each training set provided a single forest RF^w from which proximity weights (1) were determined for each of the 100 test cases; this provided a separate forest RF^p for prediction on each individual test case.

Data model	Distribution of predictors	Simulation function
1	$U(0,\,1)$	$Y = X_1 + \epsilon$
2	$U(0,\,1)$	$Y = \sin(4\pi X_1) + \epsilon$
3	$U(0,\,1)$	$Y = 10\sin(\pi X_1 X_2) + 20(X_3 - 0.5)^2 + 10X_4 + 5X_5 + \epsilon$
4	$U(0,\,1)$	$Y = \frac{X_1}{X_2 + 0.1} + \epsilon$
5	$U(0,\,1)$	$Y = e^{2X_1 X_2 + X_3} + \epsilon$
6	$\chi^2(1)$	$Y = X_1 + \epsilon$
7	$Exp(1)$	$Y = X_1 + \epsilon$
8	$Exp(1)$	$Y = X_1^2 + \epsilon$
9	$Exp(1)$	$Y = \log(X_1) + X_2 X_3 + \epsilon$

Table 7: Description of the real datasets used to evaluate prediction performance via cross-validation. Datasets 10 to 16 are available at the UCI data repository (http://archive.ics.uci.edu/ml/). For every dataset, 100 cases (2/3 of all observations if we have less than 100) were randomly selected as the training set, and the remaining cases were used as the test set. The training set was used to implement both standard RFs and CSRFs, and mean prediction errors were then calculated for each test set and averaged over 1000 simulations.

Dataset	Name	No. of predictors	No. of observations
10	Forest Fire	12	517
11	Concrete compressive strength	8	1030
12	Auto MPG	7	398
13	Automobile	19	201
14	Servo	4	167
15	House Price	13	506
16	Computer Hardware	7	209

This provides some evidence that the approach of targeted predictions in CSRFs generally improves the "one-size-fits-all" standard implementation of RF for predictions.

While fixed choices of MTN^w in the CSRF method generally improve upon the standard RF implementation, an appendix describes some approaches for automating the selection of MTN^w for defining the proximity weights (17). In repeating the numerical studies, the resulting predictions also turn out to better than standard RF and also very similar to the best performance obtained from a fixed tuning parameter setting.

Table 8: Mean prediction errors by case-specific random forests (CSRFs). For the standard RFs ($MTN^w = \infty$), the number of trees per forest and the maximal terminal nodesize are 100 and 5. For the CSRFs, the number of trees per forest and the maximal terminal nodesize for RF^w are 1000 and 5, respectively; these two parameters for RF^p are 100 and 5 in order to match the standard RF implementation. The prediction errors are calculated based on 1000 simulation runs for each of data models 1 through 9 (see Table 1) and 1000 random dataset partitions into training and test sets for each of datasets 10 through 16 (see Table 2).

Data model/dataset	Averaged Prediction MSE CSRF with different MTN^w				
	5	10	20	30	∞
1	0.0298	0.0291	0.0285	0.0288	0.0301
2	0.352	0.345	0.344	0.356	0.366
3	11.222	10.964	10.678	10.746	11.228
4	0.737	0.715	0.691	0.711	0.766
5	2.197	2.138	2.095	2.141	2.262
6	0.806	0.785	0.782	0.835	0.900
7	0.385	0.377	0.370	0.394	0.426
8	9.574	9.654	9.578	10.216	10.897
9	3.221	3.163	3.099	3.143	3.248
10	4413	4473	4630	4665	4621
11	102.9	97.1	90.0	88.2	94.0
12	10.84	10.70	10.40	10.27	10.43
13	6.47×10^6	6.47×10^6	6.49×10^6	6.37×10^6	7.33×10^6
14	0.548	0.552	0.572	0.597	0.745
15	19.45	19.02	18.35	18.20	19.80
16	2373	2201	2181	2133	3080

4 Case-Specific Variable Importance

Here we describe an approach for assessing the relative importance of regressor variables in a CSRF prediction at a regressor point $\boldsymbol{X}_0$ of interest, where the importance of variables (as with the prediction itself) can depend on $\boldsymbol{X}_0$. That is, we propose a resulting *case-specific variable importance* (CSVI) measure when one has a particular target $\boldsymbol{X}_0$ for prediction. In order to do so, it is helpful to briefly recall the mechanics of the variable importance measure in the original RF method.

In the standard RF, (Breiman 2001b) proposed a heuristic way to define variable importance as follows: if, within each resampled tree, randomly permuting the values of a certain predictor variable harms the prediction for cases not included in a given tree construction, then this variable is deemed important. This notion is embodied in the "variable importance measure" of the R package *randomForest* (Liaw and Wiener 2002), which assigns highest values to variables with the greatest discrepancy between original prediction performance and prediction performance after permutation. In regression problems, this prediction performance is computed by tree-wise determining squared errors for predicting training cases left out of the resampled tree construction (known in the RF literature as out-of-bag cases) and then averaging all such errors over all trees; the process is explained in more detail below. The resulting

variable importance values have no meaning on an absolute scale, but their relative sizes can be useful for comparing across different predictor variables. To summarize the construction of variable importance measures in the standard RF implementation, suppose we have a training set $\boldsymbol{\mathcal{C}} = \{C_i,\, i = 1, \ldots,\, N\}$ of N cases $C_i = (\boldsymbol{X}_i,\, Y_i)$.

1. A standard RF is grown based on $\boldsymbol{\mathcal{C}}$, consisting of B trees each generated from a bootstrap resample of $\boldsymbol{\mathcal{C}}$.

   (a) For the bth tree, $b = 1, \ldots, B$, let $\boldsymbol{\mathcal{C}}_b^{in}$ and $\boldsymbol{\mathcal{C}}_b^{oob}$ be the subset of $\boldsymbol{\mathcal{C}}$ included and not included in the tree construction, respectively.

   (b) Predict the out-of-bag training cases (i.e, $Y_i, i \in \boldsymbol{\mathcal{C}}_b^{oob}$) using the bth tree to get $\hat{Y}_i^b$ for $i \in \boldsymbol{\mathcal{C}}_b^{oob}$.

   (c) Randomly permute the jth predictor variable values among the cases in $\boldsymbol{\mathcal{C}}_b^{oob}$ and repeat this process P times. Denote the pth permuted set $\boldsymbol{\mathcal{C}}_b^{oob}$ for the jth variable as $\boldsymbol{\mathcal{C}}_{b,p,j}^{oob}$, $p = 1, \ldots, P$. Predict each permuted case in $\boldsymbol{\mathcal{C}}_{b,p,j}^{oob}$ using bth tree to get $\hat{Y}_i^{b,p,j}$.

2. If there are M_i trees grown without the ith case ($i = 1, \ldots, N$), then its out of bag prediction by the forest without permutation is

$$\hat{Y}_i = \frac{1}{M_i} \sum_{\{b | i \in \boldsymbol{\mathcal{C}}_b^{oob}\}} \hat{Y}_i^b,$$

   and its out-of-bag prediction by the forest with the pth permutation of the jth predictor variable is

$$\hat{Y}_i^{p,\,j} = \frac{1}{M_i} \sum_{\{b | i \in \boldsymbol{\mathcal{C}}_b^{oob}\}} \hat{Y}_i^{\,b,p,j}, \quad p = 1, \ldots, P.$$

3. The prediction mean square error for the ith case (without permutation) is $Err_i = (\hat{Y}_i - Y_i)^2$, and the prediction mean square error for the ith case (with jth predictor permuted) is

$$Err_i^j = \frac{1}{P} \sum_{p=1}^{P} (\hat{Y}_i^{p,\,j} - Y_i)^2.$$

4. Define $\Delta Err_i^j = \max(0,\, Err_i^j - Err_i)$. Then the variable importance for the jth predictor by standard RFs is then

$$V_j = \frac{1}{N} \sum_{i \in \boldsymbol{\mathcal{C}}} \Delta Err_i^j$$

   and one can further normalize the variable importance values for all predictors to make them sum

to 1 as

$$VI_j = \frac{V_j}{\sum_k V_k}. \tag{18}$$

We would again like to define a similar, though case-specific, variable importance (CSVI) measure for the CSRF method. However, when one has a particular target regressor setting $\boldsymbol{X}_0$ in mind, the mechanics of the variance importance measure in standard RFs above are impractical to replicate within the CSRF method itself because one would ideally like to predict only the "hold-out" or out-of-bag cases which are "close" to $\boldsymbol{X}_0$ for any tree and such cases can be sparse (leading to unreliable, empirical prediction summaries). Viewed another way, the approach in the original RF method does not discriminate among out-of-bag cases in forming prediction measures. With this in mind, we propose a modification to the variable importance measures in standard RFs to obtain CSVI measures. Instead of averaging the prediction errors after permutations over all the out-of-bag training cases, we propose using a weighted average of these values to represent the variable importance

$$CSV_j = \sum_{i \in \boldsymbol{\mathcal{C}}} D_i \cdot \Delta Err_i^j,$$

where D_i is the proximity measure between $\boldsymbol{X}_0$ and ith training case in $\boldsymbol{\mathcal{C}}$, as defined in (17), and then we further define

$$CSVI_j = \frac{CSV_j}{\sum_k CSV_k} \tag{19}$$

to normalize these values. The relevance of (19) is that we consider the jth predictor variable as important only if its permutation harms the prediction of the training cases in close proximity to $\boldsymbol{X}_0$.

We illustrate this idea by a simple simulation example. Suppose we have a training sample of 200 iid observations $(X_1,\, X_2,\, X_3,\, Y)$ with X_1, X_2 and X_3 from independent uniform $U(0,\, 1)$ and $Y|(X_1,\, X_2,\, X_3) \sim N(5X_2 \cdot I_{X_1 \leq 0.5} - 5X_3 \cdot I_{X_1 > 0.5},\, 0.1^2)$, where I is an indicator function that takes the value 1 when the condition in its subscript is true and is 0 otherwise. In this example, the standard RF method produces the predictor variable importance values for $(X_1,\, X_2,\, X_3)$ as (0.8, 0.1, 0.1), on average. Suppose we have two target regressor cases $\boldsymbol{X}_{01} = \{X_1 = 0.25,\, X_2 = x_2,\, X_3 = x_3\}$ and $\boldsymbol{X}_{02} = \{X_1 = 0.75,\, X_2 = x_2,\, X_3 = x_3\}$, where the values x_2 of X_2 and x_3 of X_3 represent $U(0, 1)$ draws, and we hope to understand the relative variable importance of X_1 X_2 and X_3 for prediction at $\boldsymbol{X}_{01}$ and $\boldsymbol{X}_{02}$. Apparently for predicting $\boldsymbol{X}_{01}$, the variable X_3 should be considered *not* important compared to X_2, while it plays a much more important role in predicting $\boldsymbol{X}_{02}$. The standard RF variable importance measure (given above) would fail to indicate such a difference. The CSVI measure defined in (19), on the other hand, gives values (0.71, 0.25, 0.04) for variables $(X_1,\, X_2,\, X_3)$ when predicting at $\boldsymbol{X}_{01}$ and values (0.71, 0.04, 0.25) when predicting at $\boldsymbol{X}_{02}$

on average. Hence, the important variables for predicting at one part of the regressor space need not be the same as the ones key for predicting at another part or even across the entirety of the regressor space, which the CSVI measure reflects.

The next section provides an illustrative example of the CSRF method and the CSVI measure applied to a real data set.

5 Data Illustration

To illustrate the CSRF methodology, we consider a prediction problem involving the sales price of a house in Ames, Iowa. Ames is a college town (home to Iowa State University) in central Iowa with population of about 59,000, about half of whom are students. A colleague of ours was interested in selg his house and predicting a sales price based on records of residential house sales (2011-2012) in the City of Ames. An accurate prediction of sales price would clearly be useful for setting an asking price and negotiating with potential buyers. A relative dataset for generating predictions is publicly available at the Ames City Assessor website *http://www.cityofames.org/index.aspx?page=492.* The complete dataset includes a variety of qualitative and quantitative variables including sales price, gross living area, year of construction, building style, number of rooms, assessed value from previous years and so on. Table 9 lists 11 predictor variables related to features potentially influencing sales price. The 739 observations with complete records of these variables in addition to sales price comprises the training set used throughout the remainder of this section.

With such a training set in hand, a standard approach would involve estimating one global equation or developing one general procedure that could be used to predict house sales throughout the city. However, our colleague naturally wanted to predict the sales price of *his* house, based on the characteristics provided in the last column of Table 9. In this prediction problem, previous transactions involving houses with similar predictor variable values probably contain more relevant information than other transactions. Our proposed CSRF method was motivated by this idea. For our colleague's house, the standard RF and the CSRF ($MTN^w = 5$) methods predicted the sale price at \$140,079 and \$142,665, respectively. Without knowledge of our predictions, our colleague made a good deal by selg his house for \$144,000, which is closer to the CSRF prediction.

As described in Section 4, the standard RF methodology assesses a predictor variable to be important by the severity to which randomly shuffg this variable degrades out-of-sample prediction over *all* cases in a dataset (Breiman 2001b). But, with interest in the variables most important for predicting the sales

Table 9: Description of the Ames House dataset. There are 6 levels for variable "Building Type": single family home, properties converted from commercial, condominium, duplex, townhouse on edges and townhouse in the middle. There are 7 levels for variable "House Style": single story finished, 1.5 stories finished, 2 stories finished, 2.5 stories finished, 2.5 stories unfinished above ground, split foyer and split level.

	Predictor Variable	Range/Number of Levels	House of Interest
1	Gross Living Area (GLA)	$461 \sim 3769\,\text{ft}^2$	1676 ft^2
2	Basement Area	$0 \sim 2968\,\text{ft}^2$	1008 ft^2
3	Lot Area	$916 \sim 184250\,\text{ft}^2$	10744 ft^2
4	Assessed Value of Land (AVL)	\$11200 $\sim$ 222200	\$37300
5	Assessed Value of Total (AVT)	\$53200 $\sim$ 822200	\$138100
6	Building Type	6 levels	Single family home
7	House Style	7 levels	1.5 stories finished
8	Year Built	$1880 \sim 2011$	1924
9	Number of Rooms above Ground	$3 \sim 14$	7
10	Number of Cars in Garage	$0 \sim 5$	1
11	School District	1 or 5	1

price of a *specific* house, the variable important measures from standard RFs may not necessarily be the most relevant. According to the case-specific variable importance (CSVI) measure defined in Section 4, the three most important predictor variables for predicting the sales price of our colleague's house are AVT (0.543), year built (0.145) and GLA (0.069). However, the variable importance measure defined in standard RFs identifies AVT (0.716), GLA (0.105) and AVL (0.051) as the three most important variables. Perhaps not surprisingly, the standard RF places a great deal of weight on the total assessed value (AVT) of a house. Because the standard RF variable importance measure treats prediction of all house sales prices as equally important, variables such as total asssessed value that generally track well with sale price across the entire dataset will be considered important by the standard RF approach. However, the CSVI does not so heavily weight AVT for this particular house among the other predictor variables, and the CSRF provides a sales price prediction with a larger discrepancy from the total assessed value compared to the standard RF. Some of relative importance of AVT shifts to the variable *year built* according to CSVI, which indicates the older age of the house of interest plays *an* important role in the CSRF prediction.

6 Conclusions

In this paper, we introduced a case-specific random forest (CSRF) method for growing random forests tailored for prediction at specific regressor settings of interest. As illustrated in the data example of Section 5 there are situations in which one may naturally begin a prediction problem with a specific case for prediction in mind, which motivates the CSRF approach. Several numerical investigations of prediction

performance, based on both simulated and real data examples, consistently and generally showed that CSRFs outperformed the standard RFs for several fixed choices of the maximal terminal node size MTN^w used to define the proximity weights (17) in the CSRF method. While automatic selection of MTN^w is not necessary to achieve improved predictions by CSRFs over standard RFs, some data-driven selection rules are described and illustrated in an Appendix. We also proposed a formulation for a case-specific variable importance (CSVI) measure and demonstrated that globally important variables for prediction (as assessed with standard RFs) can be quite different from locally important variables relevant for predicting a specific regressor case of interest. The CSRF methodology and the CSVI measures can improve prediction and provide insight when the scientific goal is prediction for a specific case.

References

Amaratunga, D., Cabrera, J., and Lee, Y.-S. (2008). Enriched random forests. *Bioinformatics*, 24(18):2010–2014.

Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588.

Borra, S. and Ciaccio, A. D. (2010). Measuring the prediction error. a comparison of cross-validation, bootstrap and covariance penalty methods. *Computational Statistics and Data Analysis*, 54:2976–2989.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, 26(2):123–140.

Breiman, L. (1996b). Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, 24(6):2350–2383.

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

Breiman, L. (2001b). Statistical modeling: The two cultures (with rejoinder). *Statistical Science*, 16(3):199–231.

Breiman, L., Friedman, J., Stone, C., and Olshen, R. (1984). *Classification and Regression Trees.* Chapman and Hall, 1st edition edition.

Buhlmann, P. and Yu, B. (2002). Analyzing bagging. *Annals of Statistics*, 30(4):927–961.

Coppersmith, D., Hong, S., and Hosking, J. (1999). Partitioning nominal attributes in decision trees. *Data Mining and Knowledge Discovery*, 3:197–217.

De'ath, G. mvpart: Multivariate partitioning, 1.3-1. *R Package Version*.

Diaz-Uriarte, R. and de AndršŠs, S. A. (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7:3–15.

Efron, B. (2004). The estimation of prediction error: Covariance penalties and cross-validation. *Journal of the American Statistical Association*, 99(467):619–642.

Efron, B. and Tibshirani, R. (1997). Improvements on cross-validation: The .632+ bootstrap method. *Journal of the American Statistical Association*, 92(438):548–555.

Freund, Y. and Schapire, R. (1996). Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156.

Friedman, J. and Hall, P. (2000). On bagging and nonlinear estimation. *Journal of Statistical Planning and Inference*, 137(3):669–683.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63:3–42.

Goldstein, B., Polley, E., and Briggs, F. (2011). Random forests for genetic association studies. *Statistical Applications in Genetics and Molecular Biology*, 10(1):1–34.

Hammann, F., Gutmann, H., Vogt, N., Helma, C., and Drewe, J. (2010). Prediction of adverse drug reactions using decision tree modeling. *Clinical Pharmacology and Therapeutics*, 88:52–59.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2nd edition edition.

Ho, T. (1995). Random decision forest. *Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC*, 14-16:278–282.

Ho, T. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844.

Ishwaran, H., Kogalur, U., Blackstone, E., and Lauer, M. (2008). Random survival forests. *The Annals of Applied Statistics*, 2(3):841–860.

Jiang, P., Wu, H., Wang, W., Ma, W., Sun, X., and Lu, Z. (2007). Mipred: classification of real and pseudo microrna precursors using random forest prediction model with combined features. *Nucleic Acids Research*, 35(2):339–344.

Kumar, M. and Thenmozhi, M. (2006). Forecasting stock index movement: A comparison of support vector machines and random forest. *Indian Institute of Capital Markets 9th Capital Markets Conference*.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

Lin, Y. and Jeon, Y. (2006). Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590.

Meinshausen, N. (2006). Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999.

Olshen, R. (2010). Remembering leo breiman. *The Annals of Applied Statistics*, 4(4):1644–1648.

Pal, M. (2003). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222.

Palmer, D., O'Boyle, N., Glen, R., and Mitchell, J. (2007). Random forest models to predict aqueous solubility. *J Chem Inf Model*, 47(1):150–158.

Segal, M. and Xiao, Y. (2011). Multivariate random forests. *WIREs Data Mining and Knowledge Discovery*, 1:80–87.

Shi, T., Seligson, D., Belldegrun, A., Palotie, A., and Horvath, S. (2005). Tumor classification by tissue microarray profiling: Random forest clustering applied to renal cell carcinoma. *Modern Pathology*, 18(4):547–557.

Siroky, D. (2009). Navigating random forests and related advances in algorithmic modeling. *Statistics Surveys*, 3:147–163.

Ward, M., Pajevic, S., Dreyfuss, J., and Malley, J. (2006). Short-term prediction of mortality in patients with systemic lupus erythematosus: Classification of outcomes using random forests. *Arthritis and Rheumatism*, 55:74–80.

Appendix

Tuning case-specific random forests

In the development of the case-specific random forest (CSRF) method from Section 2 and in its numerical investigation in Section 3, we have seen that MTN^w is a tuning parameter that defines the

Table 10: Predictions by CSRFs using tuning algorithms 1 & 2 (denoted A1, A2) to select MTN^w values.

Data model	Overall Prediction MSE		Data model	Overall Prediction MSE	
	Tuned A1	Tuned A2		Tuned A1	Tuned A2
1	0.0289	0.0289	*9*	3.140	3.132
2	0.345	0.347	*10*	4484	4523
3	10.802	10.805	*11*	89.6	90.4
4	0.704	0.704	*12*	10.39	10.40
5	2.134	2.133	*13*	6.45×10^6	6.48×10^6
6	0.786	0.790	*14*	0.553	0.553
7	0.371	0.374	*15*	18.47	18.67
8	9.602	9.617	*16*	2296	2196

proximity weights (17) used in Step 1 of the CSRF Algorithm. While simulations in Section 3 (Table 8) have shown that CSRFs generally perform better than the standard RFs for several fixed MTN^w choices within a given data example, the best fixed choice of the tuning parameter MTN^w for CSRFs can depend on the data-generating model or even the regressor point $\boldsymbol{X}_0$ at which prediction is sought. While a variety of fixed choices of MTN^w can improve predictions over standard RFs, it is also possible to select this tuning parameter in an automatic way, as explained in the following.

A1. Common MTN^w value for every test case

For obtaining importance weights for implementing CSRFs, one could attempt to find a data-driven selection of a single MTN^w value to be used for predicting every test case, where MTN^w is selected from some set of candidate nodesizes $\mathcal{M}$ that includes ∞ (the standard RF). This can be achieved through leave-one-out cross validation as described in Tuning Algorithm 1 below. Using this algorithm, we obtained one MTN^w value for each individual simulation run with the data model/examples listed in Tables 6 and 7. Applying the CSRF method with these data-driven choices of MTN^w produced the prediction errors shown in Table 10 (labeled Tuned A1). The performance of the data-tuned CSRF procedure under this algorithm was always close to the best performing CSRF based on a fixed MTN^w choice in Table 8.

Tuning Algorithm 1 (N-fold cross validation for tuning CSRFs with one common MTN^w value)

1. For each training case $C_i = (\boldsymbol{X}_i, Y_i)$ (treating X_i as the desired point of prediction), apply the CSRF Algorithm (Section 2) to the reduced dataset $\boldsymbol{\mathcal{C}}^{(-i)} = \{C_j \in \boldsymbol{\mathcal{C}} : j \neq i\}$ for each MTN^w value in a prespecified candidate set $\mathcal{M} = \{i_1, \ldots, i_M, \infty\}$ that includes integers $i_1 < \ldots < i_M$ and

∞. For case C_i, the CSRF method with $MTN^w = m$ will give a prediction $\hat{Y}_{i,\,m}$ and a prediction error $Err_{i,\,m}$ (i.e., $Err_{i,\,m} = (y_i - \hat{y}_{i,\,m})^2$ or $Err_{i,\,m} = |y_i - \hat{y}_{i,\,m}|$ depending on error criterion used).

2. The chosen tuning parameter is then

$$\widehat{MTN}^w = \arg \min_{\{m \in \mathcal{M}\}} \sum_{i=1}^{N} Err_{i,\,m}.$$

A2. Case-specific tuning of MTN^w values

While the CSRF procedure of Appendix Section A1 (i.e., Tuning Algorithm 1) uses weighted resampling to grow trees with a data-driven choice of MTN^w, the value of MTN^w is not tuned to a particular point $\boldsymbol{X}_0$ in the regressor space where a prediction may be of interest. This may be reasonable when many predictions are sought across the regressor space, and in this situation, the resulting CSRF method generally outperforms standard RF predictions. However, given a training set $\boldsymbol{\mathcal{C}} = \{C_1, \ldots, C_N\}$ and a particular prediction point $\boldsymbol{X}_0$, we may introduce a data-driven way to select an appropriate MTN^w value for predictions at $\boldsymbol{X}_0$ by CSRF, which (as in Tuning Algorithm 1) is selected from some set of candidate nodesizes $\mathcal{M}$ that includes ∞ (the standard RF). This procedure is described in the following algorithm.“

Tuning Algorithm 2 (Case-specific tuning of MTN^w value)

1. For each training case $C_i = (\boldsymbol{X}_i, Y_i)$ (treating $\boldsymbol{X}_i$ as the desired point of prediction), apply the CSRF Algorithm (Section 2) to the reduced dataset $\boldsymbol{\mathcal{C}}^{(-i)} = \{C_j \in \boldsymbol{\mathcal{C}} \,:\, j \neq i\}$ to obtain CSRFs with different MTN^w values from a prespecified candidate set $\mathcal{M} = \{i_1, \ldots, i_M, \infty\}$ of integers $i_1 < \ldots < i_M$ and ∞. For C_i, the CSRF method with $MTN^w = m$ will give a prediction $\hat{Y}_{i,\,m}$ and a prediction error $Err_{i,\,m}$ for each $m \in \mathcal{M}$.

2. Grow another standard RF (MTN=5) using the training set $\mathcal{R}$ to find the proximity probability weights, $D_1^*, \ldots, D_N^*$, based on proximity measures between $\boldsymbol{X}_0$ and all training cases C_i, $i = 1, \ldots, N$.

3. Then the data-driven selection for MTN^w at $\boldsymbol{X}_0$ is

$$\widetilde{MTN}_0^w = \arg \min_{\{m \in \mathcal{M}\}} \left\{ \sum_{i=1}^{N} D_i^* \times Err_{i,\,m} \right\}$$

4. Proceed with the CSRF Algorithm and grow a CSRF for predicting at $\boldsymbol{X}_0$ using $MTN^w = \widetilde{MTN}_0^w$.

For every data example listed in Table 6 and 7, we re-ran CSRFs with MTN^w selected by Tuning Algorithm 2. The results from this case-specific tuning of MTN^w (labeled Tuned A2 in Table 10) were very similar to those from the CSRF approach with Tuning Algorithm 1 (Tuned A1) in these examples.

CHAPTER 5. CASE-SPECIFIC ESTIMATION OF EXPECTED PREDICTION LOSS

Abstract

Expected prediction loss for a statistical method may vary over test cases with different predictor variable values. Given a statistical method, one may be particularly interested to know how well this method may predict the response of a test case with a specified predictor variable value. For a general fitted prediction rule $\hat{f}$, existing methods all deal with estimation of the expected prediction loss in a non-case-specific manner. In this paper, we propose a method to estimate expected prediction loss case-specifically, by combining random forest methodology with an additional bootstrap. We examine the performance of our method for predictions with a linear and non-linear data-generating model. In both examples, the results show that our method produces good estimation of case-specific expected prediction loss averaged over the joint distribution of training set, which provides case-specific parallels to the non-case-specific prediction findings of others.

1 Introduction

Accurate evaluation of the performance of a learning method is a fundamental problem in model selection. This performance is usually assessed by measuring expected prediction loss on an independent test set that was not used to fit the model. Estimation of expected prediction loss is particularly relevant when a large training sample is not available (Borra and Ciaccio 2010; Hastie et al. 2009) . There are two main streams of methods for expected prediction loss estimation: 1) methods based on covariance penalties including Mallow's C_p (1973), Akaike's information criterion (1973), and Stein's unbiased risk estimate (1981), and 2) methods related to bootstrap and cross-validation techniques (Efron 2004; Efron and Tibshirani 1997; Shao 1993). Unlike covariance penalty methods, bootstrap and cross-validation based methods are nonparametric and therefore, are widely used to assess model performance in general settings.

Throughout the paper we use the following notation. Suppose the response variable Y and a p-dimensional predictor variable $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)$ are connected by a general unknown function $f(\cdot)$ through $Y = f(\boldsymbol{X}) + \epsilon$, where ϵ is a random variable with mean 0. We let $\boldsymbol{\mathcal{C}} = \{C_i \equiv (\boldsymbol{X}_i, Y_i),\ i = 1, \ldots, n\}$ represent a training dataset, where $C_1, \ldots, C_n$ are IID random vectors from a joint distribution $\mathcal{P}$. The function $f(\cdot)$ is estimated by $\boldsymbol{\mathcal{C}}$, and the estimator $\hat{f}_{\boldsymbol{\mathcal{C}}}(\boldsymbol{X}_0)$ is used to predict Y_0 for an independent test case $C_0 \equiv (\boldsymbol{X}_0, Y_0)$ drawn from the training data distribution $\mathcal{P}$.

There are different forms of expected prediction loss one can measure (Borra and Ciaccio 2010 2010). Given a loss function $L(\cdot)$, a quantity that reflects the prediction performance averaging over the distribution of test case C_0 is

$$Err(\boldsymbol{\mathcal{C}}) \quad = \quad E_{\boldsymbol{X}_0} E_{Y_0|\boldsymbol{X}_0}[L(Y_0,\, \hat{f}_{\boldsymbol{\mathcal{C}}}(\boldsymbol{X}_0)) \,|\, \hat{f}_{\boldsymbol{\mathcal{C}}},\, \boldsymbol{X}_0], \tag{20}$$

where the subscripts for the expectation operation indicate the distribution over which the expectations are taken. Expression (20) corresponds to the scenario where a fixed training set $\boldsymbol{\mathcal{C}}$ is given and one wants to know how well this prediction rule $\hat{f}_{\boldsymbol{\mathcal{C}}}$ works, without considering its sampling distribution or the joint distribution of $\boldsymbol{\mathcal{C}}$. Another expected loss often considered is

$$\overline{Err} \quad = \quad E_{\boldsymbol{\mathcal{C}}} Err(\boldsymbol{\mathcal{C}}). \tag{21}$$

Instead of conditioning on $\boldsymbol{\mathcal{C}}$, Expression (21) takes the sampling variation of $\boldsymbol{\mathcal{C}}$ into consideration by averaging $Err(\boldsymbol{\mathcal{C}})$ in (20) over the joint distribution of $\boldsymbol{\mathcal{C}}$. It should be clear that $Err(\boldsymbol{\mathcal{C}})$ and $\overline{Err}$ measure two different kinds of expected prediction loss. There a rich literature discussing both quantities (Hastie et al. 2009; Borra and Ciaccio 2010).

In some inference settings, it may be of most interest to predict a specific test case response with a given regressor variable $\boldsymbol{X}_0 = \boldsymbol{x}_0$. As an example, a home seller may be most interested in predicting the sales price of her home based on some of its characteristics $\boldsymbol{x}_0$. If the seller, say, fits a regression model using previous sales data across a city, she could get an idea of how much her house is worth, as well as assess her model's overall prediction performance by estimating (20) or (21) with well established cross-validation or bootstrap methods. Under this scenario, however, the performance of a prediction rule $\hat{f}_{\boldsymbol{\mathcal{C}}}$at a particular $\boldsymbol{x_0}$ is more meaningful than the one averaged over the joint distribution $\mathcal{P}$ of $(\boldsymbol{X}_0,\, Y_0)$. Instead of estimating (20) and (21), we may perfer to consider a test case-specific expected prediction loss quantity given by

$$Err(\boldsymbol{\mathcal{C}},\, \boldsymbol{x}_0) \quad = \quad E_{Y_0|\boldsymbol{X}_0=\boldsymbol{x}_0}[L(Y_0,\, \hat{f}_{\boldsymbol{\mathcal{C}}}(\boldsymbol{x}_0)) \,|\, \hat{f}_{\boldsymbol{\mathcal{C}}}] \tag{22}$$

or

$$\overline{Err}(\boldsymbol{x}_0) \quad = \quad E_{\boldsymbol{\mathcal{C}}} Err(\boldsymbol{\mathcal{C}}, \boldsymbol{x}_0). \tag{23}$$

Note that (22) reflects the prediction performance for a fixed test case $\boldsymbol{x}_0$ by a prediction rule fitted from a fixed training sample $\boldsymbol{\mathcal{C}}$. Expression (23) is the expected value of (22) averaged over the joint distribution of $\boldsymbol{\mathcal{C}}$. We refer to these the two quantities as the *conditional case-specific expected prediction loss* (CCSEPL, in 22) and the *mean case-specific expected prediction loss* (MCSEPL, in 23). Both the CCSEPL and MCSEPL, while important, have received little attention in the literature, in sharp contrast to their non-case-specific counterparts in (20) or (21).

In this paper we propose a method to estimate (23), by utilizing the powerful prediction performance of random forest methodology (Breiman 2001a). We organize the manuscript as follows. In Section 2, we describe the estimation procedure. In Section 3, we examine the performance of our method using simulation examples. In Section 4, we further examine the performance of our estimator with real datasets.

2 Estimation of Case-Specific Expected Prediction Loss

The bootstrap is a general resampling method for assessing expected prediction loss for a statistical method. References on this topic can be found in (Efron and Tibshirani 1997, Hastie et al. 2009). For a training dataset $\boldsymbol{\mathcal{C}}$ of size n, we get bootstrap resamples $\boldsymbol{\mathcal{B}}_1, \boldsymbol{\mathcal{B}}_2, \ldots, \boldsymbol{\mathcal{B}}_M$, where each dataset $\boldsymbol{\mathcal{B}}_i$ is obtained by resampling n cases from $\boldsymbol{\mathcal{C}}$ independently and with replacement. We denote $\boldsymbol{\mathcal{O}}_{-i}$ as the set of indices of the bootstrap resamples which do not contain the ith training case C_i. On average the size of $\boldsymbol{O}_{-i}$ is $0.368M$ for a large training set $\boldsymbol{\mathcal{C}}$ (Efron and Tibshirani 1997). An estimator of expected prediction loss (20) or (21) is given by

$$\widehat{Err} = \quad \tfrac{1}{n} \quad \sum_{i=1}^{n} \frac{1}{||\boldsymbol{\mathcal{O}}_{-i}||} \sum_{m \in \boldsymbol{\mathcal{O}}_{-i}} L(y_i, \hat{f}^*_m(\boldsymbol{x}_i)), \tag{24}$$

where $\hat{f}^*_m(\cdot)$ is the prediction rule fitted based on bootstrap resample $\boldsymbol{\mathcal{B}}_m$. The number of unique elements in a typical bootstrap resample is less than n, which makes (24) tend to overestimate the expected prediction loss. Efron and Tibshirani (1997) further improved this estimator by defining

$$\widehat{Err}_{0.632} \quad = \quad 0.368 \cdot \widehat{\overline{err}} + 0.632 \cdot \widehat{Err}, \tag{25}$$

where $\widehat{err}$ is the training sample expected prediction loss estimator,

$$\widehat{err} \quad = \quad \frac{1}{n}\sum_{i=1}^{n} L(y_i,\, \hat{f}_{\boldsymbol{\mathcal{C}}}(\boldsymbol{x}_i)).$$

Both $\widehat{Err}$ and $\widehat{Err}_{0.632}$ aim to estimate the conditional expected loss $Err(\boldsymbol{\mathcal{C}})$ in (20), but have been shown as typically good estimators only for the expected prediction loss $\overline{Err}$ in (21) (Efron 2004; Efron and Tibshirani 1997; Hastie et al. 2009; Borra and Ciaccio 2010).

The target of this paper is to evaluate the performance of a prediction rule on a specific data point $\boldsymbol{x}_0$. Such expected loss for two cases with predictor variable values and response values in close proximity to each other should be similar. In a high-dimensional dataset, it is challenging to determine the distance between two cases, because this distance should be defined with consideration of both predictor variables, but also potentially the response variable as well (Hinneberg et al. 2000). Euclidean distance on predictor variables will fail in this scenario.

Random forest (RF) methodology is a useful nonparametric data mining technique for both regression and classification problems (Breiman 2001a). RFs have been shown to have good prediction performance for high dimensional datasets, without the necessity of variable selection and pre-specification of a parametric function form. Lin and Jeon (2006) showed that the RF method is fundamentally an adaptive k-nearest neighbor method. Its algorithm partitions the regressor space into disjoint hyperrectangles and tends to group similar cases together into terminal nodes, and thereby uses nearest neighbors for prediction. In the light of this property of RFs, we propose a method to estimate case-specific expected prediction loss based on the bootstrap technique and RF method. This is an adaptation of (24) intended to modify the bootstrap estimator of expected prediction loss to a case-specific setting.

Algorithm 1

Given a training set $\boldsymbol{\mathcal{C}} = \{C_i \equiv (\boldsymbol{x}_i,\, y_i),\, i = 1,\, \ldots,\, n\}$, we fit a prediction rule $\hat{f}_{\boldsymbol{\mathcal{C}}}(\cdot)$. We want to estimate the expected prediction loss of $\hat{f}_{\boldsymbol{\mathcal{C}}}$ for an independent test case $C_0 \equiv (\boldsymbol{X}_0,\, Y_0)$ with $\boldsymbol{X}_0 = \boldsymbol{x}_0$.

1) Get M bootstrap resamples of $\boldsymbol{\mathcal{C}}$, $\boldsymbol{\mathcal{B}}_1$, $\boldsymbol{\mathcal{B}}_2$, $\ldots$, $\boldsymbol{\mathcal{B}}_M$ and obtain a prediction rule $\hat{f}^*_m$ using $\boldsymbol{\mathcal{B}}_m$ as the training set, $m = 1,\, \ldots,\, M$.

2) Define $\boldsymbol{\mathcal{O}}_{-i}$ is as in Expression 24. Get the out-of-bag estimated loss for the ith case $C_i \equiv (\boldsymbol{X}_i,\, Y_i)$ in $\boldsymbol{\mathcal{C}}$ as

$$\widetilde{Err}_i \quad = \quad \frac{1}{||\boldsymbol{\mathcal{O}}_{-i}||} \sum_{m \in \boldsymbol{\mathcal{O}}_{-i}} L(y_i,\, \hat{f}^*_m(\boldsymbol{x}_i)),\, i = 1,\, \ldots,\, n,$$

where $||\boldsymbol{\mathcal{O}}_{-i}||$ is the size of set $\boldsymbol{\mathcal{O}}_{-i}$.

3) Grow a RF, namely RF_{err} using $\widetilde{Err}_i$, $i = 1, \ldots, n$ as the responses and $\boldsymbol{x}_i$, $i = 1, \ldots, n$ as predictor variables.

4) Input $\boldsymbol{X} = \boldsymbol{x}_0$ to RF_{err} to get the estimated expected prediction loss $\widehat{Err}_0 = RF_{err}(\boldsymbol{x}_0)$ for C_0 by the prediction method used to obtain $\hat{f}_{\boldsymbol{\mathcal{C}}}(\boldsymbol{x}_0)$.

The expected prediction loss estimation method in Algorithm 1 is a modification of the approach to expected loss estimation in of (21) (Efron 2004, Hastie et al. 2009), except that here we utilize a RF to estimate the expected loss on a specific test case C_0 with $\boldsymbol{X} = \boldsymbol{x}_0$. We may also adjust this estimator using the training expected loss estimator, similar to $\widehat{Err}_{0.632}$ in (25) (Efron and Tibshirani 1997). This is shown in Algorithm 2.

Algorithm 2

1) Do steps 1) and 2) in Algorithm 1 to get $\widetilde{Err}_i$, $i = 1, \ldots, n$.

2) Fit $f(\cdot)$ using the training set $\boldsymbol{\mathcal{C}}$, and calculate the training expected loss estimator for each case, $\widetilde{Err}_{i,\,training} = L(y_i,\, \hat{f}_{\boldsymbol{\mathcal{C}}}(\boldsymbol{x}_i))$. Then the corrected expected loss estimator for C_i is

$$\widetilde{Err}_{i,\,0.632} = 0.368 \cdot \widetilde{Err}_{i,\,training} + 0.632 \cdot \widetilde{Err}_i.$$

3) Grow a RF, namely RF_{err} using $\widetilde{Err}_{i,\,0.632}$, $i = 1, \ldots, n$ as the responses and $\boldsymbol{x}_i$, $i = 1, \ldots, n$ as predictor variables.

4) Input $\boldsymbol{X} = \boldsymbol{x}_0$ to RF_{err} to get the estimated expected prediction loss $\widehat{Err}_0 = RF_{err}(\boldsymbol{x}_0)$ for C_0 by the prediction method used to obtain $\hat{f}_{\boldsymbol{\mathcal{C}}}(\boldsymbol{x}_0)$.

3 Simulation Studies

3.1 A Linear Model Example

We examine the performance of our CSEPL methods (Algorithms 1 and 2) using a simulation example. The predictor variable is $\boldsymbol{X} = (X_1, \ldots, X_5)$, where $X_1, \ldots, X_5$ are IID $U(0, 1)$ and the response variable $Y \mid \boldsymbol{X} = \boldsymbol{x}$ is $N(f(\boldsymbol{x}), 0.1^2)$ with

$$f(\boldsymbol{x}) \quad = \quad x_1 + 5x_2. \tag{26}$$

A training set $\boldsymbol{\mathcal{C}}$ containing 1000 independent observations and a test set $\boldsymbol{\mathcal{H}}$ containing 50 independent observations were randomly generated from this joint distribution. We fit a linear model by assuming $f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{5} \beta_j x_j$. Our purpose is to estimate the expected prediction loss for $\hat{f}_{\boldsymbol{\mathcal{C}}}$ on each test case.

Recall that the routine expected loss estimation methods based on bootstrap and the cross-validation aim at conditional expected loss (20) (on training set $\boldsymbol{\mathcal{C}}$), but have been shown to be good estimators only for the mean expected prediction loss (21) (Hastie et al. 2009), where both expected loss (20) and (21) are non-case-specific. Here we investigate whether Algorithm 1 and 2 are good estimators of conditional expected loss or mean expected loss for individual test cases (i.e., CCSEPL in (22) or MCSEPL in (23)). A test case C_0 was predicted using $\hat{f}_{\boldsymbol{\mathcal{C}}}$ and the squared deviation of prediction was calculated by $(y_0 - \hat{f}_{\boldsymbol{\mathcal{C}}}(\boldsymbol{x}_0))^2$. Its expected loss estimation $\widehat{Err}_0$ was obtained using Algorithm 1 and 2. This process was repeated 1000 times, with the covariate values for test cases $\boldsymbol{x}_0$ held unchanged, so that we can possibly calculate expected prediction loss case-specifically over simulations. The training set $\boldsymbol{\mathcal{C}}$ was either held constant or randomly generated in each simulation, in order to examine how well our our bootstrap based algorithms estimate conditional expected prediction loss (22) (on training set $\boldsymbol{\mathcal{C}}$) or mean expected prediction loss (23). These two results are shown in Figure 6 and Figure 7, respectively.

In Figure 6 and Figure 7, the average of the expected loss estimates (black), the averaged loss estimates (red) and the 5% and 95% quantiles of the loss estimates from 1000 simulations (blue) are shown for each of the 50 test cases. Both algorithms failed to give satisfactory estimates of CCSEPL in (22), when the training set $\boldsymbol{\mathcal{C}}$ was fixed over simulations. Only about half of the 90% estimate (blue) successfully included the average of the expected loss estimates (black), although the estimated loss over all test cases were both close to the averaged expected loss, which should not be surprising given the previous results by others (Borra and Ciaccio 2010; Efron and Tibshirani 1997) (results not shown). There is not much difference between Algorithm 1 and 2, based respectively on the usual bootstrap estimator and the 0.632

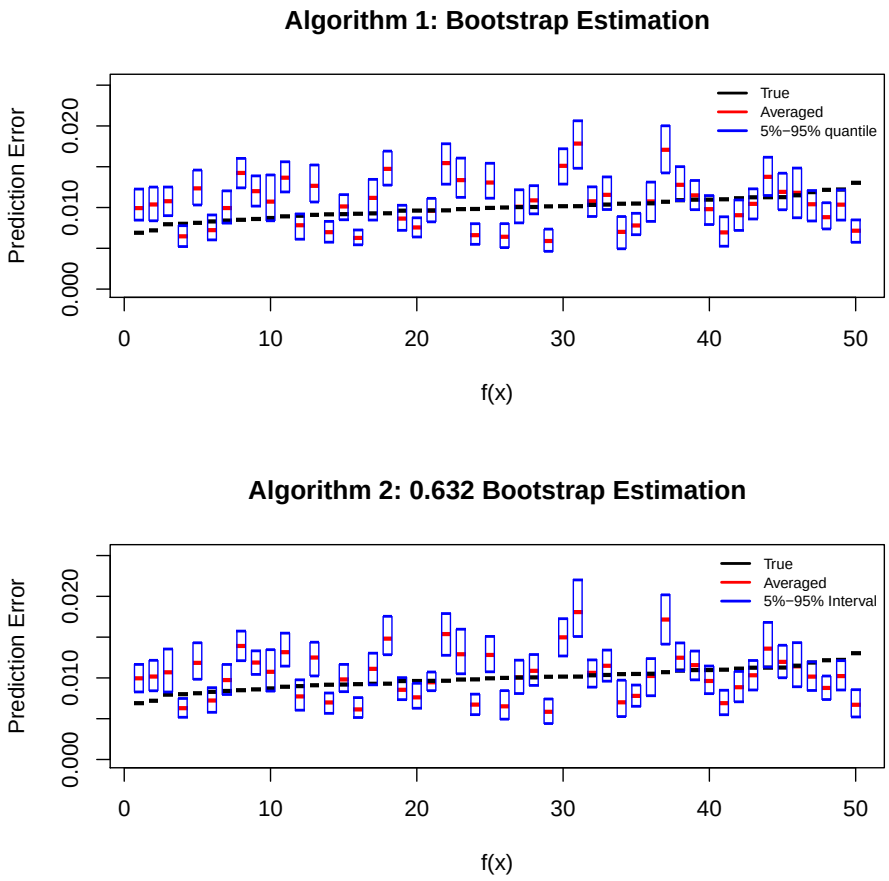


Figure 6: Estimation of conditional case-specific expected prediction loss (linear model in (26)).

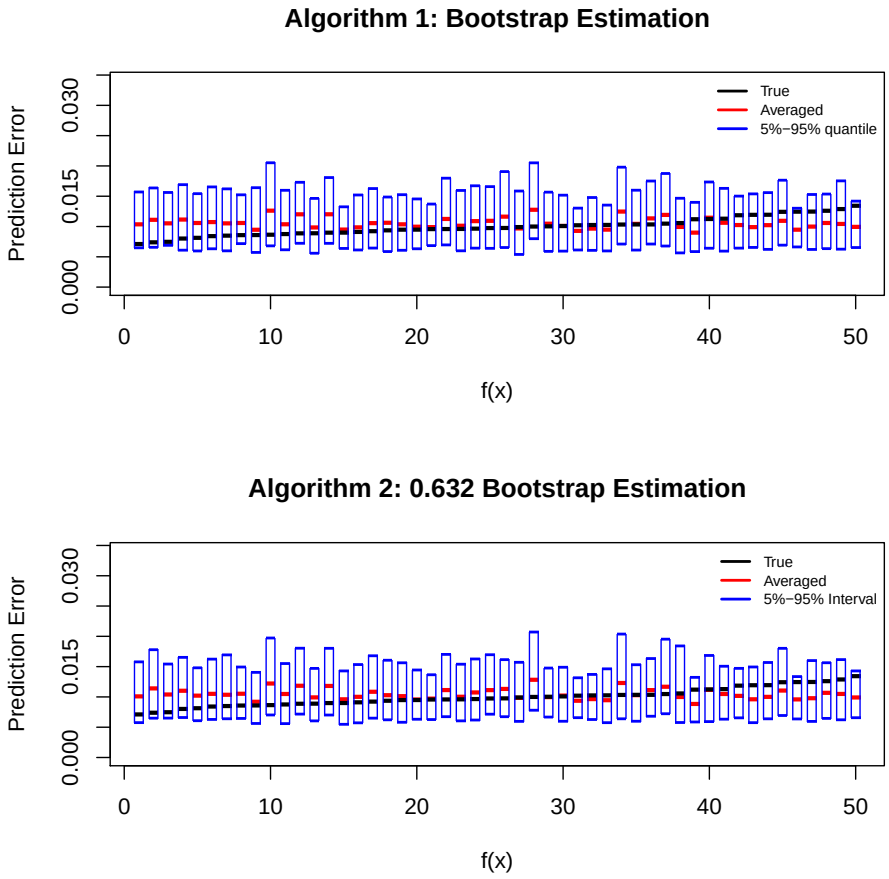


Figure 7: Estimation of mean case-specific expected prediction expected loss (linear model in (26)).

bootstrap estimator (Efron and Tibshirani 1997). However, both algorithms estimated MCSEPL in (23) very well in that the averaged expected losses (black) have little deviations from the averaged estimates (red) and from the 5% and 95 quantiles of the loss estimates (blue) (Figure-7).

3.2 A Nonlinear Model Example

We further test the performance of our expected prediction loss estimation methods on nonlinear models using *Friedman #1*, a simulation example that has been used in (Friedman 1991, Breiman 2001a, Breiman 2001b, Zhang 2012). Here we used RFs as the method or role for fitting the nonlinear model f. Predictor variables $\boldsymbol{X} = (X_1, \ldots, X_{10})$ are IID from independent $U(0, 1)$ distribution and the response variable $Y \mid \boldsymbol{X} = \boldsymbol{x}$ is from $Y = f(\boldsymbol{x}) + N(0, 3^2)$, with

$$f(\boldsymbol{x}) \quad = \quad 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5. \tag{27}$$

A training set $\boldsymbol{\mathcal{C}}$ of size 1000 and a test set $\boldsymbol{\mathcal{H}}$ of size 50 were randomly generated from this joint distribution of $(\boldsymbol{X}, Y)$. RFs were grown with $\boldsymbol{\mathcal{C}}$ and expected prediction loss were calculated for individual test cases in each of 1000 simulations. As in the linear model example of Subsection 3.1 we considered scenarios with $\boldsymbol{\mathcal{C}}$ held constant or not.

Figure 8 indicates that both algorithms generated expected loss estimates with less variation (blue intervals and red lines) than the averaged expected loss over test cases (black lines). This led to large estimation biases. As in the linear model example, Algorithm 1 and 2 performed well in estimating MCSPE by RF methodology (Figure-9). The biases were largely reduced. Again the difference between the two algorithms was small.

4 Discussion

Other authors have discovered that expected prediction loss estimation methods based on the bootstrap work well only for evaluating the mean prediction performance of a rule $\hat{f}$ over the joint distribution of training set $\boldsymbol{\mathcal{C}}$ Hastie et al. 2009; Efron and Tibshirani 1997; Efron 2004. In the case-specific setting and associated errors (22) and (23), our case-specific error estimation methods agree with this fact (i.e. MCSEPL in (23) is better than CCSEPL in (22)), as shown in Section 3.

In both Algorithms 1 and 2, we utilize the fact that RFs are k-nearest neigbor prediction rules (Lin and Jeon 2006), and assume RFs can do an acceptable job of estimating expected prediction loss of a

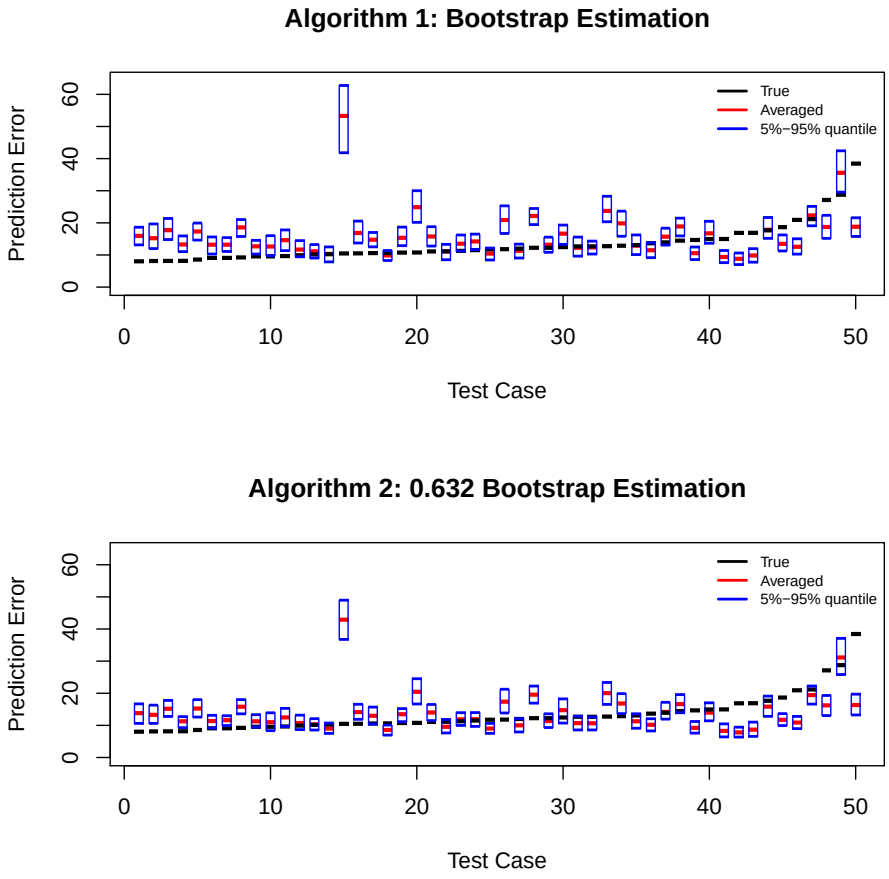


Figure 8: Estimation of conditional case-specific expected prediction loss (non-linear model in (27)).

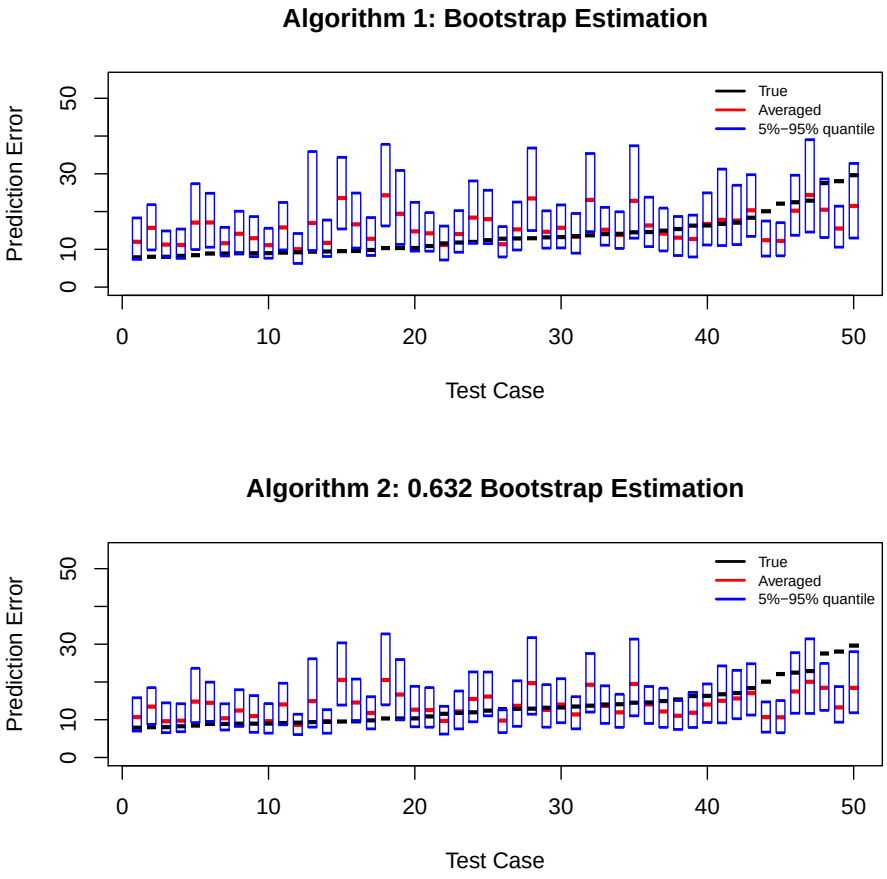


Figure 9: Estimation of mean case-specific expected prediction loss (non-linear model in (27)).

specific test case C_0 by assigning some training cases in $\boldsymbol{\mathcal{C}}$ as the neighbors of C_0. Because data are sparse in high dimensional problems, estimation of case-specific expected loss can be very difficult for a problem with many predictor variables or a complicated function relationship between Y and $\boldsymbol{X}$, $f(\cdot)$, because a test case of interest may not have any "near neighbors". In Subsection 3.2, we applied the RF method to the *Friedman #1* simulation example, where Y was associated with 10 predictors through a complex nonlinear function. A training set containing 1000 cases is still sparse for this dimensionality. To examine this, we repeated the same simulation procedure as in Subsection 3.2 with a non-fixed training set $\boldsymbol{\mathcal{C}}$. However, here the covariates values of 50 training cases were always fixed, which are exactly the same as the covariate values of the 50 test cases across the 1000 simulations. This guarantees that every test case has at least a close neighbor.

Figure 10 shows the expected prediction loss estimates for all 50 test cases which have the same covariates values with 50 training cases in $\boldsymbol{\mathcal{C}}$. Compared to Figure 9, the estimation is much better now that we have guaranteed each test case has a near neighbor. The sample correlation values between the average of the expected loss estimates (black lines) and the averaged loss estimates (red lines) by Algorithm 1 and 2 are 0.80 and 0.81, respectively, in contrast to 0.49 and 0.51 in the simulation study in Figure 9. It is worth noting that the 0.632 bootstrap estimator (Algorithm 2) gave loss estimates with smaller variance than the routine bootstrap estimator (Algorithm 1).

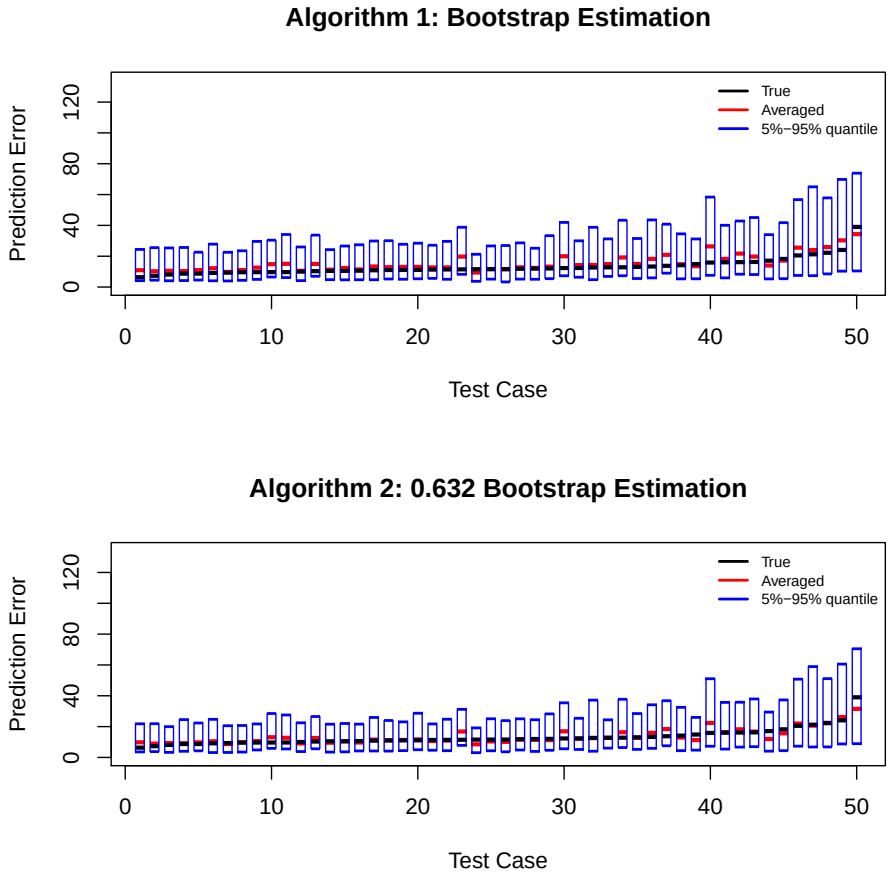


Figure 10: Estimation of case-specific expected prediction loss (non-linear in model (27)). The covariate values of all test cases are present in the training set to guarantee each test case has at least one close neighbor.

References

Amaratunga, D., Cabrera, J., and Lee, Y.-S. (2008). Enriched random forests. *Bioinformatics*, 24(18):2010–2014.

Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588.

Borra, S. and Ciaccio, A. D. (2010). Measuring the prediction error. a comparison of cross-validation, bootstrap and covariance penalty methods. *Computational Statistics and Data Analysis*, 54:2976–2989.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, 26(2):123–140.

Breiman, L. (1996b). Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, 24(6):2350–2383.

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

Breiman, L. (2001b). Statistical modeling: The two cultures (with rejoinder). *Statistical Science*, 16(3):199–231.

Breiman, L., Friedman, J., Stone, C., and Olshen, R. (1984). *Classification and Regression Trees.* Chapman and Hall, 1st edition edition.

Buhlmann, P. and Yu, B. (2002). Analyzing bagging. *Annals of Statistics*, 30(4):927–961.

Coppersmith, D., Hong, S., and Hosking, J. (1999). Partitioning nominal attributes in decision trees. *Data Mining and Knowledge Discovery*, 3:197–217.

De'ath, G. mvpart: Multivariate partitioning, 1.3-1. *R Package Version*.

Diaz-Uriarte, R. and de AndrŠŠs, S. A. (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7:3–15.

Efron, B. (2004). The estimation of prediction error: Covariance penalties and cross-validation. *Journal of the American Statistical Association*, 99(467):619–642.

Efron, B. and Tibshirani, R. (1997). Improvements on cross-validation: The .632+ bootstrap method. *Journal of the American Statistical Association*, 92(438):548–555.

Freund, Y. and Schapire, R. (1996). Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156.

Friedman, J. and Hall, P. (2000). On bagging and nonlinear estimation. *Journal of Statistical Planning and Inference*, 137(3):669–683.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, 63:3–42.

Goldstein, B., Polley, E., and Briggs, F. (2011). Random forests for genetic association studies. *Statistical Applications in Genetics and Molecular Biology*, 10(1):1–34.

Hammann, F., Gutmann, H., Vogt, N., Helma, C., and Drewe, J. (2010). Prediction of adverse drug reactions using decision tree modeling. *Clinical Pharmacology and Therapeutics*, 88:52–59.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2nd edition edition.

Ho, T. (1995). Random decision forest. *Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC*, 14-16:278–282.

Ho, T. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844.

Ishwaran, H., Kogalur, U., Blackstone, E., and Lauer, M. (2008). Random survival forests. *The Annals of Applied Statistics*, 2(3):841–860.

Jiang, P., Wu, H., Wang, W., Ma, W., Sun, X., and Lu, Z. (2007). Mipred: classification of real and pseudo microrna precursors using random forest prediction model with combined features. *Nucleic Acids Research*, 35(2):339–344.

Kumar, M. and Thenmozhi, M. (2006). Forecasting stock index movement: A comparison of support vector machines and random forest. *Indian Institute of Capital Markets 9th Capital Markets Conference.*

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

Lin, Y. and Jeon, Y. (2006). Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, 101(474):578–590.

Meinshausen, N. (2006). Quantile regression forests. *Journal of Machine Learning Research*, 7:983–999.

Olshen, R. (2010). Remembering leo breiman. *The Annals of Applied Statistics*, 4(4):1644–1648.

Pal, M. (2003). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222.

Palmer, D., O'Boyle, N., Glen, R., and Mitchell, J. (2007). Random forest models to predict aqueous solubility. *J Chem Inf Model*, 47(1):150–158.

Segal, M. and Xiao, Y. (2011). Multivariate random forests. *WIREs Data Mining and Knowledge Discovery*, 1:80–87.

Shi, T., Seligson, D., Belldegrun, A., Palotie, A., and Horvath, S. (2005). Tumor classification by tissue microarray profiling: Random forest clustering applied to renal cell carcinoma. *Modern Pathology*, 18(4):547–557.

Siroky, D. (2009). Navigating random forests and related advances in algorithmic modeling. *Statistics Surveys*, 3:147–163.

Ward, M., Pajevic, S., Dreyfuss, J., and Malley, J. (2006). Short-term prediction of mortality in patients with systemic lupus erythematosus: Classification of outcomes using random forests. *Arthritis and Rheumatism*, 55:74–80.

GENERAL CONCLUSIONS

Random forest (RF) is an "off-the-shelf" widely used machine learning method that shows competitive prediction performance. In this dissertation, we studied some characteristics of RFs, and tried to generalize the methodology and improve its prediction.

In CHAPTER 2, we found that out-of-sample prediction errors can be decreased when the regressors are augmented with independent predictor variables to generate random forests. This phenomenon can be encountered for both classification and regression problems. Using a simple simulated example, we concluded that the independent predictor augmentation spread the weights on training cases when predicting a test case, which alleviates the overfitting in some datasets. We gave two real data examples and showed that this phenomenon is not rare in data analysis.

In CHAPTER 3, we applied Breiman's iterative debiasing approach to regression trees and presented this bias correction method can reduce prediction errors in RFs dramatically. We compared this bias correction method with the default bias correction method in the R package *randomForest*, and found the new method often outperformed the default one. We discussed identifying the optimal method (uncorrected RFs, RFs with default bias correction, and iterative debiasing RFs with different number of iterations) using cross-validation.

In CHAPTER 4, we proposed a new approach to generate RFs specific to a test case of interest, which differs from the standard way of growing RFs. Examples of simulations and real datasets showed that the Case-Specific RFs (CSRFs) almost always outperformed the standard RFs. We further defined the Case-Specific Variable Importance (CSVI) measure to reflect the relative importance of predictor variable regarding the prediction of a particular test case.

In CHAPTER 5, we proposed a method to estimate expected prediction loss on a pre-specified regressor point, by combining RF methodology and the routine bootstrap application in prediction error estimation. Simulated examples showed that our method can well estimate the case-specific expected prediction loss averaging over the joint distribution of training sample.

ACKNOWLEDGEMENT

Ames is such a peaceful and lovely small town, where I spent one of the most important and joyful periods of time in my life. I am grateful for the flexibility Iowa State University granted me to study what I am interested in and work on what I am good at. In Snedecor Hall, I have known many intelligent and talented people who have provided great help to me in the past few years.

I would like to thank my major professor, Dr. Dan Nettleton, for his persistent support and guidance during my whole graduate study in Department of Statistics. Dan was always able to describe complex problems and explain abstract concepts clearly with simple words, largely benefiting my understanding and thinking. I was offered a couple of data analysis opportunities collaborating with scienctists and engineers outside statistics, which contributed a lot to building up my resume. Conversations with Dan on my furture plan were very beneficial to my career.

I want to express my gratitude to my comajor professor, Dr. Dan Nordman, for his insightful advice on my Ph.D. dissertation. Dan's solid knowledge in statistical theories was indispensible in processes of idea formation, problem solving and paper writing.

I feel fortunate for the chances of learning from my favorite teacher, Dr. Stephen Vardeman. Steve's classes were often challenging and "time-consuming" but always informative and enlightening. His ability to grasp new statistical methods overnight really amazed me.

I would like to thank my POS commitee Dr. Ken Koehler and Dr. Huaiqing Wu, for their helpful questions and discussions in my preliminary exam, defense and dissertation. I am grateful to Drs. Philip Dixon, Dean Isaacson and Max Morris for answering my statistics questions and confusions.

I would like to thank my families, friends and classmates at Snedecor Hall for their understanding and help.

Last, I owe my gratitude to a special parameter, who is always there to consistently give me inner peace and calm, overcome difficulty and variance, and endlessly back me up in every aspect of my life.