Bagging and Random Forests

Xiaogang Su, Ph.D.
Department of Mathematical Sciences
University of Texas at El Paso (UTEP)
xsu@utep.edu

April 24, 2018

Contents

1 Introduction

In the next two topics, we are going to study Perturb and Combine (P&C) predictive modelling methods that built upon tree models, including bagging, random forests, and boosting. These methods have been devised to take advantage of the instability of trees to create models that are more powerful.

1.1 Perturb and Combine

Perturb and combine (P&C) methods generate multiple models (called *base learners*) by manipulating the distribution of the data or altering the construction method and then integrating the results (Breiman, 1998).

The attractiveness of P&C methods is their improved performance over single models. Bauer and Kohavi (1999) demonstrated the superiority of P&C methods with extensive experimentation. One reason why simple P&C methods give improved performance is variance reduction, according to Breiman. If the base models have low bias and high variance, then averaging decreases the variance. In contrast, combining stable models could negatively affect performance.

Any unstable modelling method can be used as base learners, but trees are most often chosen because of their speed and flexibility. Common 'perturb'ing methods used in P&C includes: resampling, subsampling, adding noise, adaptively reweighting, or randomly choosing from the competitor splits.

An *ensemble model* is the combination of multiple models. The combinations can be formed by majority voting on the classifications, weighted voting where some models have more weight, or averaging (weighted or unweighted) of the predicted values.

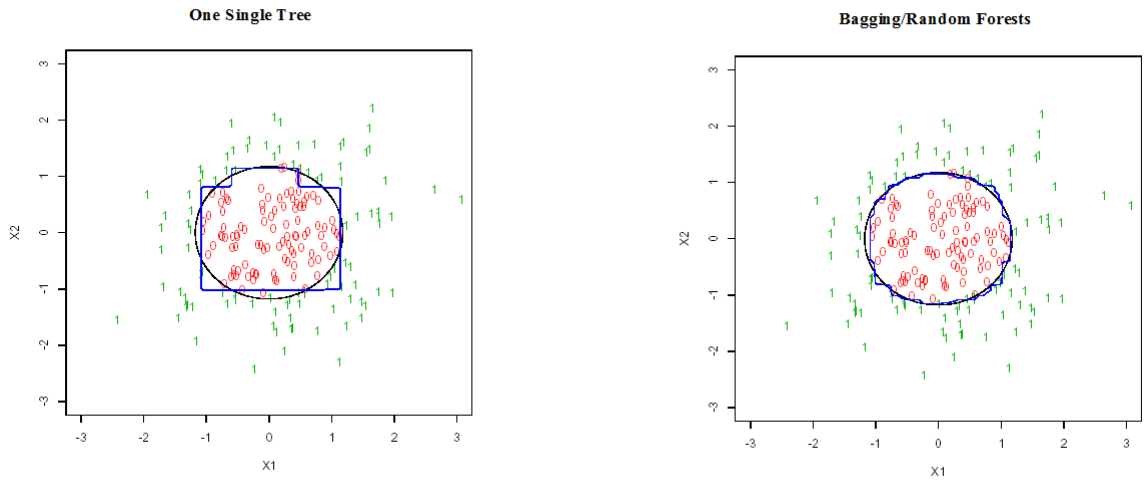

Figure 1: Comparison of the Classification Margin Provided by One Single Tree vs. Ensemble Method.

Figure 1 illustrates the classification margin provided by one single tree model vs. model ensemble method. Ensemble methods are a very active area of research in the field of machine

learning and statistics. Many other P&C methods have been devised and their empirical and theoretical properties have been explored.

1.2 Bootstrap

The *bootstrap* resampling method is most commonly used in P&C methods. To motivate, suppose that we are asked to estimate the interquartile range (IQR) of house prices in a city. Give learning sample data $\mathcal{L} = (x_1, \ldots, x_n)$, where x_i denotes the price of the ith house, the sample IQR can be computed. To assess reliability, we want to find the standard error of IQR computed from a sample and possible construct a 95% confidence interval. Compared to confidence intervals for the mean house price, this is a much more difficult statistical inference problem. However, it can be conveniently solved with the bootstrap method. We generate B bootstrap $\{\mathcal{L}_b : b = 1, \ldots, B\}$, each of size n, from $\mathcal{L}$ by sampling with replacement. From each bootstrap sample $\mathcal{L}_b$, we compute IQR_b, for $b = 1, \ldots, B$. Now we end up with B bootstrap values of IQR, which can then be used for calculating all the quantities of interest (e.g., standard deviation, confidence intervals).

The bootstrap resampling method was introduced by Bradley Efron in 1979. It was named from the phrase "to pull oneself up by one's bootstraps", which is widely believed to come from "*the Adventures of Baron Munchausen*" – a 1988 British adventure fantasy comedy film. The method was popularized in 1980s due to the introduction of computers in statistical practice. It has a strong mathematical background. It is well known as a method for estimating standard errors, bias, and constructing confidence intervals for parameters.

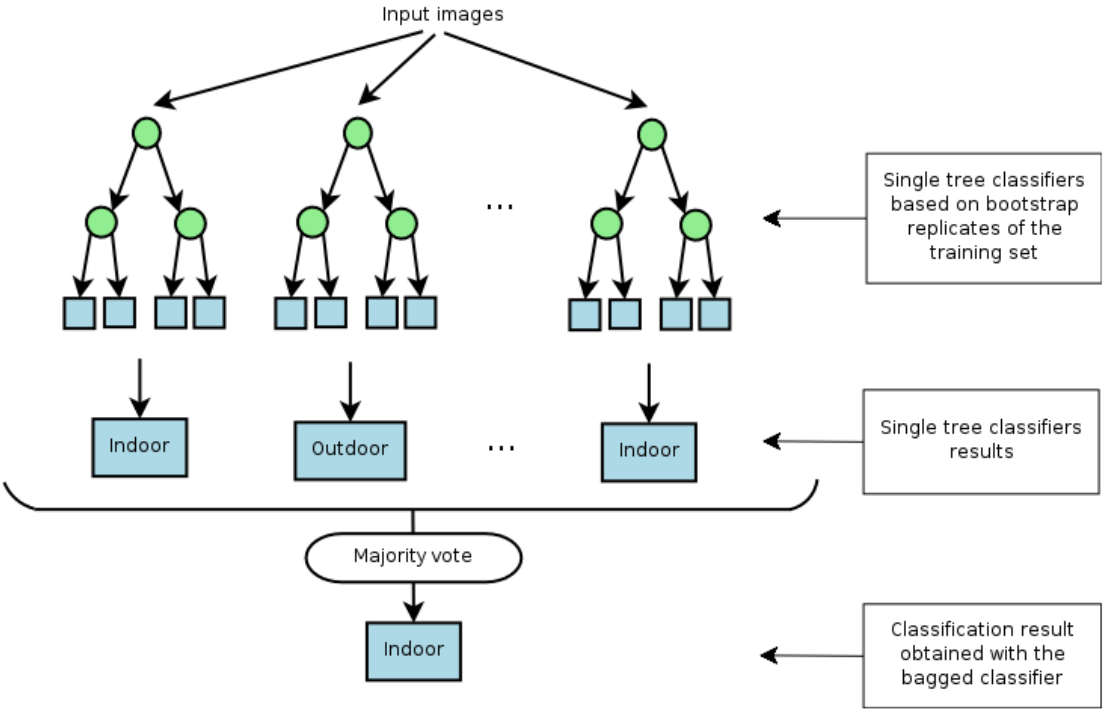

Figure 2: Illustration of Bagging.

Heuristically, we use bootstrap distribution as a way to estimate the variation in a statistic

based on the original data. There are several distribution concepts involved here. Recall that, when we take multiple samples from the population, we compute a statistic from each sample. The distribution of the values of this statistic is called the *sampling distribution*. In bootstrapping, we take multiple bootstrap samples from the original sample and we compute a sample statistic from each bootstrap sample. These values form the *bootstrap distribution*. We are hoping that the bootstrap distribution mimics the sampling distribution well.

The bootstrap does not replace or add to the original data. Note that there are n^n possible distinct bootstrap samples. The average number of distinct observations in each bootstrap sample is about $0.632\,n$. In other words, roughly 37% of the examples in the training set $\mathcal{L}$ do not appear in a particular bootstrap training set $\mathcal{L}_b$. This part of the data, denoted as $\mathcal{L}_{(b)}$ is called the 'out-of-bag' (OOB) sample by Breiman (1996) and supplies a natural cross-validation set for $\mathcal{L}_b$.

2 Bagging

"Bagging" stands for "**b**ootstrap **agg**regat**ing**", first proposed by Breiman (1996). It is an ensemble method for combining multiple model predictions.

2.1 Bagging Algorithm

Given training data $\mathcal{L} = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$, we want to make prediction for an new observation with predictor values equal to $\mathbf{x}$. Bagging first sample B data sets $\{\mathcal{L}_b : b = 1, \ldots, B\}$, each consisting of n observations randomly selected from $\mathcal{L}$ with replacement. These bootstrap samples form B quasi replicated training sets. Then we train a machine or model on each $\mathcal{L}_b$ for $b = 1, \ldots, B$ and obtain a sequence of predictions $\{\hat{y}_1(\mathbf{x}), \ldots, \hat{y}_B(\mathbf{x})\}$ based on $\mathbf{x}$. The final aggregate classifier can be obtained by averaging in regression) or majority voting in classification. The specific algorithm is given in Algorithm 1.

Data: Data $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ for classification or regression and a new predictor vector $\mathbf{x}_0$ only.
Result: out-of-bag prediction accuracy measure and predicted value $\hat{y}_0$ at $\mathbf{x}_0$.
initialize B – # bootstrap samples and V – # folds for cross-validation (CV);
begin
  for $b \leftarrow 1$ *to* B **do**
    Draw a bootstrap sample $\mathcal{L}_b$ from $\mathcal{L}$;
    Obtain the corresponding OOB sample $\mathcal{L}'_b = \mathcal{L}/\mathcal{L}_b$;
    Grow a large initial tree with $\mathcal{L}_b$ and prune;
    Identify the best tree $\mathcal{T}_b^\star$ with V-fold CV ;
    Based on $\mathcal{T}_b^\star$, predict $\hat{y}_{ib}$ for every $i \in \mathcal{L}'_b$ and $\hat{y}_{0b}$ for $\mathbf{x}_0$.
  end
  Aggregate $\hat{y}_0 = \frac{1}{B}\sum_{b=1}^B \hat{y}_{0b}$ and the prediction accuracy measure.
end

Algorithm 1: Boostrap Aggregating (Bagging)

Note that the OOB-based predictions for observations in the original sample $\mathcal{L}$ can be used to evaluate the bagging method. For example, one may obtain OOB estimates of the prediction/classification errors, which has been shown by Breiman (1996) close to evaluation based on an independent test sample.

There are several variants of the re/sub-sampling methods in bagging, including: 1) the "Standard" bagging where each of the B bootstrap resamples has size n and created with replacement; 2) Sub-bagging that creates subsamples of size $< n$; 3) no replacement – same as bagging or sub-bagging, but using sampling without replacement; 4) Overlap vs. non-overlap: Should the B subsamples overlap? i.e. create n' subsamples each with n' training data.

2.2 R Implementation

In R, bagging is implemented in the `ipred` package. Essentially, the following four functions are involved. The argument `nbagg=` specifies an integer giving the number of bootstrap replications. `coob=T` is a logical indicating whether an out-of-bag estimate of the error rate (misclassification error, root mean squared error or Brier score = $(1/n)\sum_{i=1}^{n}(\hat{p}_i - y_i)^2$) should be computed. See '`predict.classbagg`' for details. The `control` argument specifies options that control details of the `rpart` algorithm, see `rpart.control`. It is wise to set `xval = 0` in order to save computing time. Note that the default values depend on the type of the response y.

```
ipredbagg.factor(y, X=NULL, nbagg=25, control=
                 rpart.control(minsplit=2, cp=0, xval=0),
                 comb=NULL, coob=FALSE, ns=length(y), keepX = TRUE, ...)
ipredbagg.numeric(y, X=NULL, nbagg=25, control=rpart.control(xval=0),
                  comb=NULL, coob=FALSE, ns=length(y), keepX = TRUE, ...)
ipredbagg.Surv(y, X=NULL, nbagg=25, control=rpart.control(xval=0),
               comb=NULL, coob=FALSE, ns=dim(y)[1], keepX = TRUE, ...)
## S3 method for class 'data.frame':
bagging(formula, data, subset, na.action=na.rpart, ...)
```

2.3 Variance Reduction

Bagging helps reduce variance with the following intuition. If each single classifier is unstable - that is, it has high variance, the aggregated classifier has a smaller variance than a single original classifier. The aggregated classifier can be thought of as an approximation to the true average obtained by replacing the probability distribution P with its bootstrap approximation obtained by concentrating mass $1/n$ at each point $(\mathbf{x}_i, y_i)$.

Breiman (1996) has the following argument for variance reduction in the regression setting. Suppose that data $\{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$ are independently drawn from a distribution P. Given fixed $\mathbf{x}$, consider the ideal aggregate estimator $f_{\text{ag}}(\mathbf{x}) = E_P \hat{f}^{\star}(\mathbf{x})$, where the bootstrap dataset consists of observations $\{(\mathbf{x}_i^{\star}, \mathbf{y}_i^{\star}) : i = 1, \ldots, n\}$ sampled from P rather than from the current

sample data. Namely, $\hat{f}^\star(\mathbf{x})$ is constructed from sampling distributions. Then we have

$$\begin{aligned} E_P\left\{y - \hat{f}^\star(\mathbf{x})\right\}^2 &= E_P\left\{y - f_{\text{ag}}(\mathbf{x}) + f_{\text{ag}}(\mathbf{x}) - \hat{f}^\star(\mathbf{x})\right\}^2 \\ &= \left\{y - f_{\text{ag}}(\mathbf{x})\right\}^2 + \left\{f_{\text{ag}}(\mathbf{x}) - \hat{f}^\star(\mathbf{x})\right\}^2 \\ &\geq E_P\left\{y - f_{\text{ag}}(\mathbf{x})\right\}^2 \end{aligned}$$

Therefore, true population aggregation never increases mean squared error for regression. The above argument does not hold for classification under 0/1 loss, because of the nonadditivity of bias and variance.

Bagging works well for "unstable" base learning algorithms, but it can slightly degrade the performance of "stable" learning algorithms. Unstable learning algorithms are those where small changes in the training set result in large changes in predictions. These includes Neural network; Decision trees and Regression trees; Subset selection in linear/logistic regression. Stable learning algorithms includes K-nearest neighbors.

Bagging helps when a learning algorithm is good on average but unstable with respect to the training set. But if we bag a stable learning algorithm, we can actually make it worse. Bagging almost always helps with regression, but even with unstable learners it can hurt in classification. If we bag a poor and stable classifier we can make it horrible.

> "The vital element is the instability of the prediction method. If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy." (Breiman, 1996)

There are also theoretical works that quantify the term "stableness" precisely. Another important factor that influences the performance of bagging is the correlation among base learners. The more uncorrelated base models often lead to the better performance of the ensemble model. Random forest was developed in this spirit.

3 Random Forest

The quite popular Random forest (RF) can be considered as refinement of bagged trees. The term came from random decision forests that was first proposed by Tin Kam Ho of Bell Labs in 1995. The method combines Breiman's "bagging" idea and Ho's "random subspace method" to construct a collection of decision trees with controlled variations.

Table 1: Test Error % in Classification of RF Compared to Boosting Reproduced from Breiman (1996). In RF 100 trees were used while in boosting 50 trees were used for aggregation.

	Data Characteristics				Test Set Errors %			
Data Set	Train size	Test size	Inputs	Classes	Adaboost	RF-m	RF-1	Tree in RF
Glass	214	—	9	6	22.0	20.6	21.2	36.9
Breast cancer	699	—	9	2	3.2	2.9	2.7	6.3
Diabetes	768	—	8	2	26.6	24.2	24.3	33.1
Sonar	208	—	60	2	15.6	15.9	18.0	31.7
Vowel	990	—	10	11	4.1	3.4	3.3	30.4
Ionosphere	351	—	34	2	6.4	7.1	7.5	12.7
Vehicle	846	—	18	4	23.2	25.8	26.4	33.1
German credit	1000	—	24	2	23.5	24.4	26.2	33.3
Image	2310	—	19	7	1.6	2.1	2.7	6.4
Ecoli	336	—	7	8	14.8	12.8	13.0	24.5
Votes	435	—	16	2	4.8	4.1	4.6	7.4
Liver	345	—	6	2	30.7	25.1	24.7	40.6
Letters	15,000	5,000	16	26	3.4	3.5	4.7	19.8
Sat-images	4,435	2,000	36	6	8.8	8.6	10.5	17.2
Zip-code	7,291	2,007	256	10	6.2	6.3	7.8	20.6
Waveform	300	3,000	21	3	17.8	17.2	17.3	34.0
Twonorm	300	3,000	20	2	4.9	3.9	3.9	24.7
Threenorm	300	3,000	20	2	18.8	17.5	17.5	38.4
Ringnorm	300	3,000	20	2	6.9	4.9	4.9	25.7

Data: Data $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and a new predictor vector $\mathbf{x}_0$.
Result: out-of-bag prediction accuracy measure and predicted value $\hat{y}_0$ at $\mathbf{x}_0$.
initialize B – # bootstrap samples and m – # predictors randomly selected at each split;
begin
  for $b \leftarrow 1$ *to* B **do**
    Draw a bootstrap sample $\mathcal{L}_b$ from $\mathcal{L}$;
    Obtain the corresponding OOB sample $\mathcal{L}'_b = \mathcal{L}/\mathcal{L}_b$;
    Grow a large initial tree $\mathcal{T}_b$ with $\mathcal{L}_b$ by evaluating randomly selected m predictors only at each split;
    Based on $\mathcal{T}_b$, predict $\hat{y}_{ib}$ for every $i \in \mathcal{L}'_b$ and $\hat{y}_{0b}$ for $\mathbf{x}_0$.
  end
  Aggregate $\hat{y}_0 = \frac{1}{B}\sum_{b=1}^B \hat{y}_{0b}$ and the prediction accuracy measure.
end

Algorithm 2: Random Forests (RF)

Random forest, very similar to bagging, is a classification and regression method based on the aggregation of a large number of tree models. Specifically, it is an ensemble of trees constructed from a training data set and internally validated to yield a prediction of the response given the predictors for future observations. For each tree grown on a bootstrap sample, the error rate for observations left out of the bootstrap sample is monitored. This is called the "out-of-bag" error rate. But there are two important different steps from bagging: (1), at each tree split, a random sample of m (called `mtry` in R implementation) features is drawn, and only those m features are considered for splitting. Typically $m = \sqrt{p}$ or $\log_2 p$, where p is the number of features. (2), neither pruning nor tree size selection is done at each step. In other words, the base learner is a large initial random tree.

The flow chart of random forests is provided in Figure 3 and pseudo-code is given in Algorithm **??**. Random forests tries to improve on bagging by "de-correlating" the trees, owing to the random selection of m predictors.

3.1 Empirical Performance of RF

The following two tables, taken from Breiman (2001) demonstrate the empirical performance of RF in regression and classification when compared to boosting and bagging.

Let's focus on the classification problem, presented in Table 1. The simplest random forest with random features is formed by selecting at random, at each node, a small group of input variables to split on. Grow the tree using CART methodology to maximum size and do not prune. The size m of the group is fixed. Two values of F were tried. The first used only one randomly selected variable, i.e., $m = 1$, denoted as RF-1. The second took m to be the first integer less than $\log_2 p + 1$, where p is the number of inputs, denoted as RF-m.

The results of these runs are given in Table 1. Column 6 are the results selected from the two group sizes m by means of lowest out-of-bag error. The 7th column is the test set error using just one random feature to grow the trees. The 8th column contains the out-of-bag estimates of the generalization error of the individual trees in the forest computed for the best setting (RF-m or RF-1). This estimate is computed by using the left-out instances as a test set in each tree grown

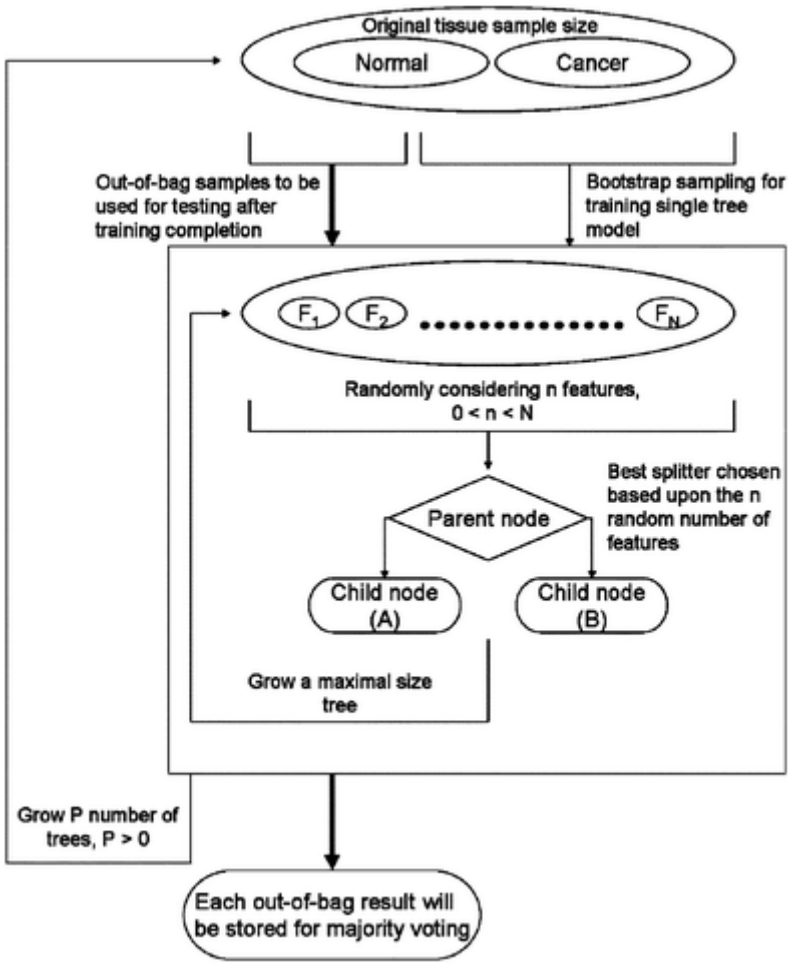

Figure 3: The Flow Chart of Random Forests.

and averaging the result over all trees in the forest.

The error rates using random input selection compare favorably with Adaboost. The procedure is not overly sensitive to the value of m. The average absolute difference between the error rate using $m = 1$ and the higher value of m is less than 1%. The difference is most pronounced on the three large data sets. It was surprising that using a single randomly chosen input variable to split on at each node could produce good accuracy.

The empirical results for regression are presented in Table 2.

3.2 Implementation and Other Features of RF

Random forests is implemented in R package `randomForest`. It has several important and intriguing features, including variable importance (VI) ranking, partial dependence plot, and proximity matrix. A list of importance functions are given below.

```
combine                 Combine Ensembles of Trees
```

Table 2: Test Error % in Regression of RF Compared to Bagging Reproduced from Breiman (2001). The adaptive bagging method was proposed in Breiman (1999) to reduce bias and operates effectively in classification as well as in regression.

	Data Characteristics			Mean-Squared Test Set Error		
Dataset	p	#Training	#Test	Bagging	Adaptive bagging	Forest
Boston Housing	12	506	10%	11.4	9.7	10.2
Ozone	8	330	10%	17.8	17.8	16.3
Servo	4	167	10%	24.5	25.1	24.6
Abalone	8	4177	25%	4.9	4.9	4.6
Robot Arm	12	15,000	5,000	4.7	2.8	4.2
Friedman#1	10	200	2,000	6.3	4.1	5.7
Friedman#2	4	200	2,000	21.5	21.5	19.6
Friedman#3	4	200	2,000	24.8	24.8	21.6

```
getTree                 Extract a single tree from a forest.
grow                    Add trees to an ensemble
importance              Extract variable importance measure
outlier                 Compute outlying measures
partialPlot             Partial dependence plot
plot.randomForest       Plot method for randomForest objects
predict.randomForest    predict method for random forest objects
randomForest            Classification and Regression with Random
                        Forest
rfImpute                Missing Value Imputations by randomForest
treesize                Size of trees in an ensemble
tuneRF                  Tune randomForest for the optimal mtry
                        parameter
varImpPlot              Variable Importance Plot
varUsed                 Variables used in a random forest
```

3.2.1 Variable Importance Ranking

Variable importance measure is an attractive feature offered by RF. Variable importance measure helps answer questions such as which features are important risk factors or predictors of the response. This issue cannot be fully addressed by simply examining the splitting variables shown in a single final tree structure, as an important variable can be completely masked by other correlated ones. The variable importance technique in random forests has been increasingly studied in its own right and applied as a tool for variable selection in various fields. This method generally belongs to the "cost-of-exclusion" (Kononenko and Hong, 1997) feature selection category, in which the importance or relevance of a feature is determined by the difference in some model performance

measure with and without the given feature included in the modeling process.

Let V_j denote the importance measure of the j-th covariate or feature X_j for $j = 1, \ldots, p$. We construct random forests by taking B bootstrap samples $\mathcal{L}_b$, $b = 1, \ldots, B$. For each tree $\mathcal{T}_b$, the b-th out-of-bag sample $\mathcal{L}'_b$ is sent down $\mathcal{T}_b$ to compute a performance measure $G(\mathcal{T}_b)$. Next, the values of the j-th covariate in OOB sample $\mathcal{L}'_b$ are randomly permuted. The permuted out-of-bag sample is then sent down $\mathcal{T}_b$ to recompute $G_j(\mathcal{T}_b)$. The relative difference between $G(\mathcal{T}_b)$ and $G_j(\mathcal{T}_b)$ is recorded. The procedure is repeated for B bootstrap samples. As a result, the importance measure V_j is the average of the relative differences over all B bootstrap samples. The whole procedure is summarized in Algorithm 3.

Data: Data $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$
Result: Variable importance measures V_j via RF
initialize B, m, and set $V_j = 0$ for $j = 1, \ldots, p$;
begin
  for $b = 1$ *to* B **do**
    Draw a bootstrap sample $\mathcal{L}_b$ from $\mathcal{L}$;
    Obtain the corresponding OOB sample $\mathcal{L}'_b = \mathcal{L}/\mathcal{L}_b$;
    Grow a large initial tree $\mathcal{T}_b$ with $\mathcal{L}_b$ by evaluating randomly selected m predictors only at each split;
    Send $\mathcal{L}'_b$ down to $\mathcal{T}_b$ and compute a performance measure $G(\mathcal{T}_b)$;
    for $j = 1$ *to* p **do**
      Permute the X_j values in $\mathcal{L}'_b$, denote the permuted OOB sample as $\mathcal{L}'_{bj}$;
      Send $\mathcal{L}'_{bj}$ down to $\mathcal{T}_b$ and compute a performance measure $G_j(\mathcal{T}_b)$;
      Compute $V_j := V_j + \dfrac{G(\mathcal{T}_b) - G_j(\mathcal{T}_b)}{G(\mathcal{T}_b)}$;
    end
  end
  Average $V_j := V_j/B$ for $j = 1, \ldots, p$.
end

Algorithm 3: Variable Importance Measures in Random Forests (RF)

The standard RF offers two different variable importance measures for each predictor. See Figure 4 in classification. The first measure is computed from permuting OOB data: For each tree, the prediction error on the out-of-bag portion of the data is recorded (error rate for classification, MSE for regression). Then the same is done after permuting each predictor variable. The difference between the two are then averaged over all trees, and normalized by the standard deviation of the differences. If the standard deviation of the differences is equal to 0 for a variable, the division is not done (but the average is almost always equal to 0 in that case). In contrast, this permutation VI measure is directly based on the prediction accuracy rather than on the splitting criterion (the second measure). It is defined as the difference between the OOB error resulting from a data set obtained through random permutation of the predictor of interest and the OOB error resulting from the original data set. Permutation of an 'important' predictor is expected to increase the OOB error, leading to a high permutation VI measure.

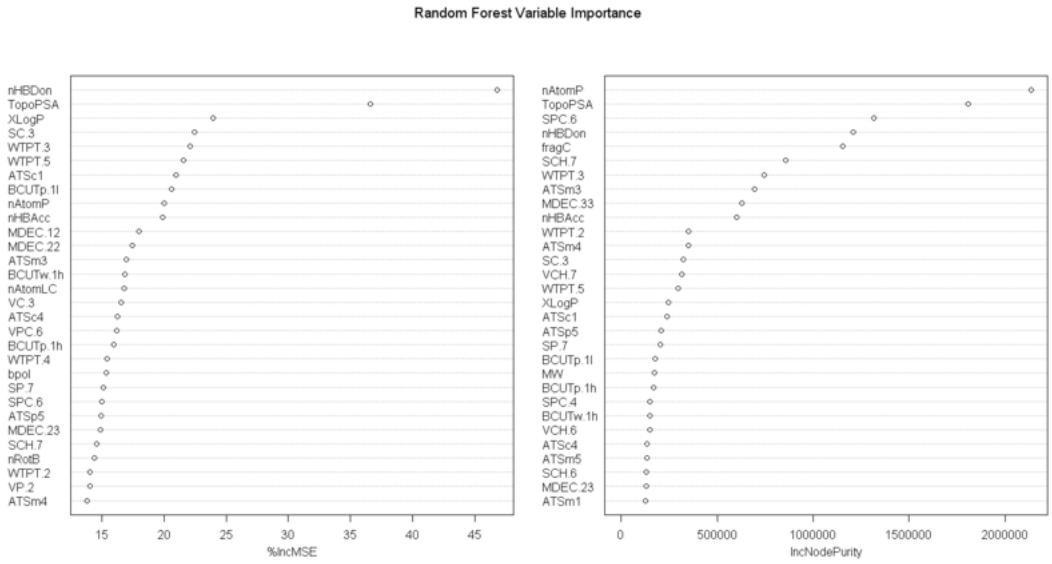

Figure 4: Variable Importance Measures and Ranking via Random Forests.

The second measure is the total decrease in node impurities from splitting on the variable, averaged over all trees. For classification, the node impurity is measured by the Gini index. For regression, it is measured by residual sum of squares. The Gini VI measure of a predictor of interest is the sum of the decrease of Gini impurity criteria of the splits that are based on this predictor, scaled by the total number of trees in the forest. An 'important' predictor is often selected for splitting and yields a high decrease of Gini impurity when selected, leading to a high Gini VI measure.

While the permutation VI measure is more frequently used in practice, the question of the choice of the VI type and the properties of these VI measures are still subjects of current research.

The number of trees in the forest should quite generally increase with the number of candidate predictors, so that each predictor has enough opportunities to be selected. If we have, say, 20,000 predictors (for instance gene expressions) in the data set, the number of trees should by no way be set to the default value of 500. Roughly speaking, it should be chosen such that two independent runs of the random algorithm yield very similar results. It is recommended to try several increasing values and to stop increasing as soon as the measures of interest (such as prediction error or VI measures) stabilize. Note that a smaller number of trees might yield the same prediction accuracy as a larger number but less reliable VI measures.

3.2.2 Partial Dependence Plot

The partial dependence plot (see Figure 5) provides a novel way to explore the overall effects of a predictor on the response and extract interpretation from RF. We briefly describe the idea in the regression setting.

Given a prediction or regression function, denoted by $h(\mathbf{x}) = h(\mathbf{x}_1, \mathbf{x}_2)$ by partitioning $\mathbf{x}$ into $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$, we want to know the relative importance or contribution of predictor $\mathbf{x}_1$ on the response. Note that this problem is also addressed by variable importance from a different angle.

Figure 5: Partial Dependence Plot via Random Forests.

For graphical exploration, the dimension of $\mathbf{x}_1$ should be low, say, up to 3. Usually, components of $\mathbf{x}_1$ come from the top of the variable importance ranking list.

The theoretical partial dependence function for $\mathbf{x}_1$ is defined as

$$h_1(\mathbf{x}_1) = E_{\mathbf{x}_2} h(\mathbf{x}_1, \mathbf{x}_2) = \int h(\mathbf{x}_1, \mathbf{x}_2) f(\mathbf{x}_2) d\mathbf{x}_2,$$

where $f(\mathbf{x}_2)$ denotes the marginal density function of $\mathbf{x}_2$. Note that $h_1(\mathbf{x}_1)$ is different from the conditional expectation

$$E\{h(\mathbf{x}_1, \mathbf{x}_2) \,|\, \mathbf{x}_1\} = \int h(\mathbf{x}_1, \mathbf{x}_2) f(\mathbf{x}_2|\mathbf{x}_1) d\mathbf{x}_2,$$

unless $\mathbf{x}_1$ and $\mathbf{x}_2$ are independent. The interpretation of partial dependence relates to marginal expectation, representing the average effect of $\mathbf{x}_1$ on $m(\mathbf{x})$ after accounting for the average effects of $\mathbf{x}_2$. However, conditioning on $\mathbf{x}_1$ in the latter quantity $E\{m(\mathbf{x}_1, \mathbf{x}_2) \,|\, \mathbf{x}_1\}$ ignores or removes the effects of $\mathbf{x}_2$.

It can be easily seen that, if $h(\mathbf{x}_1, \mathbf{x}_2)$ is either additive $h(\mathbf{x}_1, \mathbf{x}_2) = g_1(\mathbf{x}_1) + g_2(\mathbf{x}_2)$ or multiplicative $h(\mathbf{x}_1, \mathbf{x}_2) = g_1(\mathbf{x}_1) g_2(\mathbf{x}_2)$, the partial dependence function $h_1(\mathbf{x}_1)$ will be able to recover $g_1(\mathbf{x}_1)$ up to some constant; however, this is not the case for the conditional $E\{h(\mathbf{x}_1, \mathbf{x}_2) \,|\, \mathbf{x}_1\}$.

In Regression, the partial dependence function $h_1(\mathbf{x}_1)$ at one point $\mathbf{x}_1$ can be the estimated as

$$\hat{h}_1(\mathbf{x}_1) = \frac{1}{n} \sum_{i=1}^{n} \hat{h}(\mathbf{x}_1, \mathbf{x}_{i2}),$$

where $\{\mathbf{x}_{i2} : i = 1, \ldots, n\}$ runs through every value of $\mathbf{x}_2$ in the training data. In classification where the response y is categorical with levels $\{1, \ldots, K\}$, the logits (i.e., the log of the fraction of votes) can be used for defining the prediction function $m(\mathbf{x})$ so that

$$h(\mathbf{x}) = \log \Pr(y = 1) - \frac{1}{K} \sum_{k=1}^{K} \log \Pr(y = k)$$

and the partial dependence function $\hat{m}(\mathbf{x}_1)$ can be computed in a similar manner.

Data: Data $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and Random Forests $\mathcal{F} = \{\mathcal{T}_b\}_{b=1}^B$
Result: Partial dependence plot for X_j
initialize Obtain equal-distanced points $\{x_{mj} : m = 1, \ldots, M\}$ from the range of X_j ;
begin
  for $m = 1$ *to* M **do**
    Replace the X_j value in $\mathcal{L}$ with x_{mj} to form new data set $\mathcal{L}_m$ of size n;
    Send $\mathcal{L}_m$ down to each tree in the random forests $\mathcal{F}$ and obtain RF-predicted values $\{\hat{y}_{imj}\}_{i=1}^n$;
    Average $\hat{y}_{mj} = \sum_{i=1}^n \hat{y}_{imj}/n$ as an estimate of $m_1(x_{mj})$
  end
  Plot points $(x_{mj}, \hat{y}_{mj})$ for $m = 1, \ldots, M$.
end

Algorithm 4: Partial Dependence Plot in Random Forests (RF)

How to make the partial dependence plot for univariate $\mathbf{x}_1$ is illustrated in Algorithm 4. The equal-distanced points from the range of X_j can be replaced with distinct values of X_j; also, percentiles of X_j can be added to the horizontal axis of the plot to show the rough distribution of X_j. Clearly making the partial dependence plot can be computationally intensive. Fortunately, such a computation with tree models can be rapidly done without reference to the data. Algorithm 4 can be readily extended to higher dimensions for exploring interactions.

3.2.3 Proximity Measures

RF introduces a novel way to define proximity $\mathbf{D} = \{d_{ii'} = d(i, i')\}$ between two observations i and i'. To do so, first initialize all proximities to zeroes. For any given tree, apply the tree to all cases. If case i and case i' both end up in the same node, increase proximity $d_{ii'}$ between i and i' by one. Then accumulate over all trees in RF and normalize by twice the number of trees in RF.

Data: Data $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and Random Forests $\mathcal{F} = \{T_b : b = 1, \ldots, B\}$.
Result: Proximity matrix $\mathbf{D} = (d_{ii'}) \in \mathbb{R}^{n \times n}$
initialize $d_{ii'} = 0$ for $i < i' = 1, \ldots, n$;
begin
  for $i = 1$ *to* n **do**
    for $i' = i$ *to* n **do**
      for $b = 1$ *to* B **do**
        if $\mathbf{x}_i$ *and* $\mathbf{x}_{i'}$ *fall in the same terminal node of* T_b **then**
          $d_{ii'} := d_{ii'} + 1$
        end
      end
    end
  end
  Average $d_{ii'} := d_{ii'}/B$;
end

Algorithm 5: Proximity Matrix in Random Forests (RF)

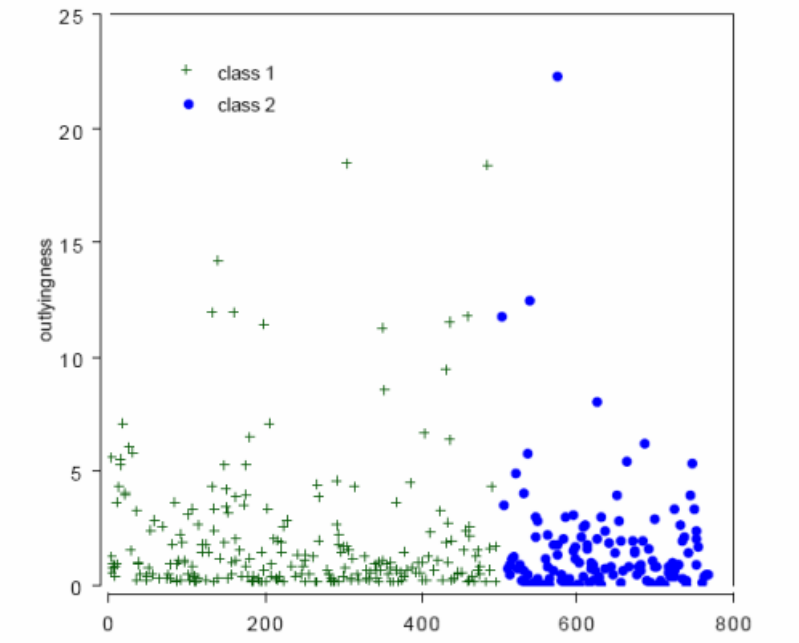

Figure 6: Outlier Detection with Proximity Measures Supplied by Random Forests.

The resulting matrix $\mathbf{D}$ of dimension $n \times n$ provides intrinsic measure of proximity, in the sense that observations that are "alike" will have proximities close to one. The closer proximity to 0, the more dissimilar cases i and i' are. Storage of $\mathbf{D}$ can resort to sparse matrices. Note that the measure is invariant to monotone transformations. Also, the measure is clearly defined for any type

of independent variables, including categorical ones.

The matrix $\mathbf{D'} = (1 - d_{ii'})$ can be treated as distances in a high dimensional space and can be used for further exploration. Some exemplary usages include informative data views using metric multidimensional scaling (MDS), missing value imputation, cluster analysis, and outlier detection (as illustrated on Figure 6). One advantage of $\mathbf{D'}$ is that it handles both continuous and categorical predictors in an unified way.

References

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

Breiman, L. (1999). Using adaptive bagging to debias regressions. Technical Report 547, Dept of Statistics, UCB.

Breiman, L. (2001). Random forests. *Machine Learning*, **45** (1): 5–32.

Breiman, L. (2001). Statistical modeling: the two cultures. *Statistical Science*, **16**(3): 199–231.

Hastie T., Tibshirani, R., and Friedman, J. H. (2008). *Elements of Statistical Learning*, 2nd Edition. Chapman & Hall.

Liaw, A. and Wiener, M. (2001). Classification and Regression by Random Forest. *R News*, vol **2/3**.