CKW

Statistics 8330: Data Analysis III
Random Forests

Suggested Reading: James, Witten, Hastie, and Tibshirani (JWHT), 2013, Chapter 8.2.2

Supplemental Reading: Hastie, Tibshirani, and Friedman (HTF), 2009, Chapter 15

Christopher K. Wikle

University of Missouri
Department of Statistics

Random Forests

Random forests, like bagging, is a bootstrap-based procedure based on a committee of learning trees. The primary difference between random forests and bagging is that the trees are effectively *decorrelated*.

The decorrelation in random forests occurs because at each split in the tree-building process (for each bootstrap sample) we select the splitting rule from a randomly selected relatively small subset of the original input variables. This allows different variables to be involved in the tree building process across the bootstrap samples, which makes the trees less dependent on each other (and dominant covariates) and hence, reduces the variance of the prediction.

Recall that this variance reduction is important with trees because they are inherently noisy, but if grown fairly deep, have relatively low bias. So, averaging many low-biased independently distributed trees together gives relatively unbiased and low variance predictions/classifications.

Random Forests (cont.)

CKW

Recall from Data Analysis I and II that the variance of an average of correlated variables is larger than the average of the same number of independent variables (i.e., the effective degrees of freedom are less for correlated data).

One can show that an average of B identically distributed random variables, each with variance σ^2, that have pairwise correlation ρ, is given by

$$\rho\sigma^2 + \sigma^2\frac{(1-\rho)}{B}.$$

Thus, as B increases, one still has limiting variance $\rho\sigma^2$. If we could reduce ρ, then we could obtain a smaller variance for the average. Hence, the goal of random forests is to reduce the inherent dependence in bagging.

Random Forests (cont.)

CKW

As mentioned, we do this by randomly selecting a smaller number of input variables, say m, where $m \leq p$; typically, one starts with $m = \sqrt{p}$ for classification and $m = p/3$ for regression. In general, the best results will depend on the specific data and this choice should be treated as a parameter to be tuned.

After we grow B such trees, say $\{T(x; \Theta_b)\}_1^B$, the random forest regression predictor is then:

$$\hat{f}_{rf}^B(x) = \frac{1}{B} \sum_{b=1}^{B} T(x; \Theta_b),$$

where Θ_b corresponds to the cut points associated with the bth tree.

Random Forests (cont.)

Consider the general random forest algorithm from HTF (2009, Alg. 15.1):

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Random Forests (cont.)

CKW

Consider the example for the spam data from HTF (2009):

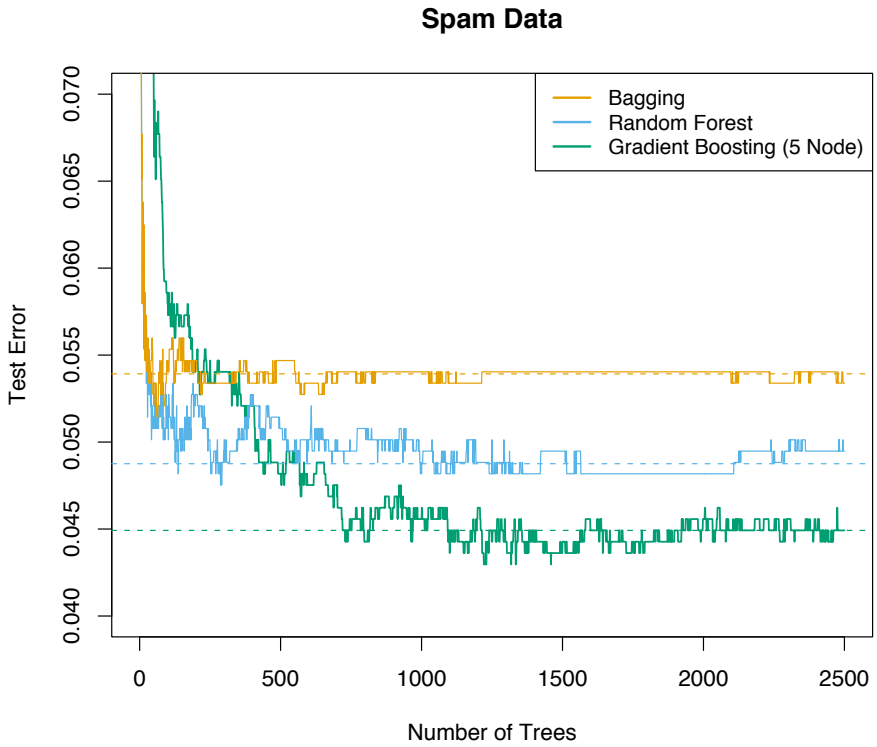

FIGURE 15.1. *Bagging, random forest, and gradient boosting, applied to the spam data. For boosting, 5-node trees were used, and the number of trees were chosen by 10-fold cross-validation (2500 trees). Each "step" in the figure corresponds to a change in a single misclassification (in a test set of 1536).*

Random Forests (cont.)

Consider the example for the California housing data from HTF (2009):

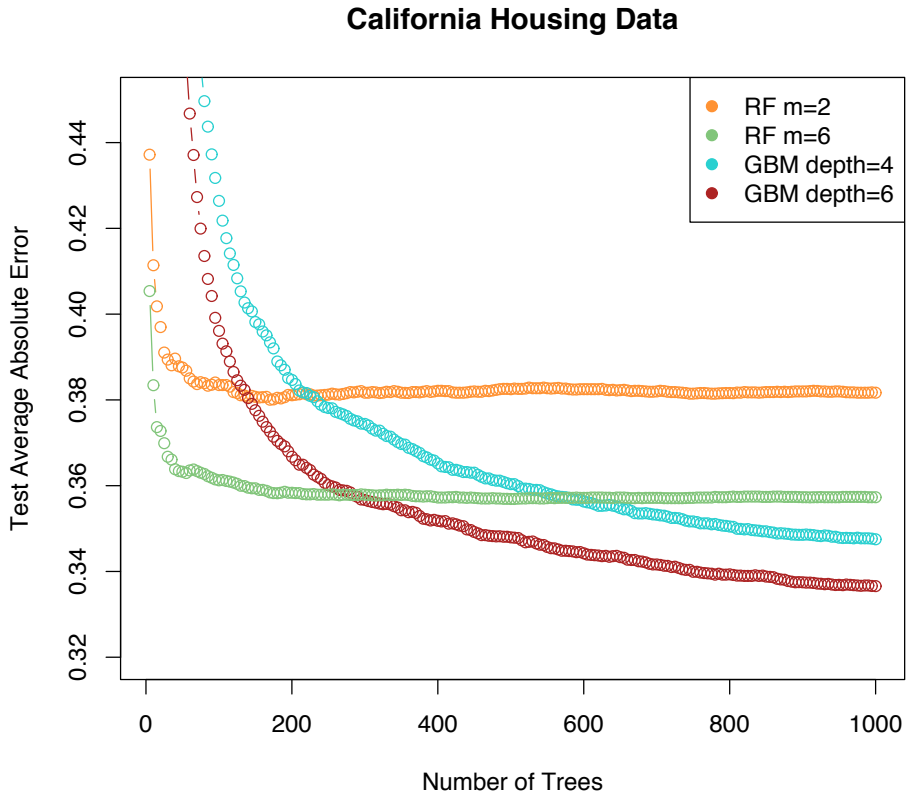

FIGURE 15.3. *Random forests compared to gradient boosting on the California housing data. The curves represent mean absolute error on the test data as a function of the number of trees in the models. Two random forests are shown, with $m = 2$ and $m = 6$. The two gradient boosted models use a shrinkage parameter $\nu = 0.05$ in (10.41), and have interaction depths of 4*

Random Forests (cont.)

CKW

In general, random forests are better than bagging in most cases and comparable to gradient boosting. The comparisons are actually difficult to do because there are parameters to choose in all of these methods, and it is difficult to investigate (i.e., tune) each method optimally (and equivalently!) to a specific data set.

In addition, which method is best depends significantly on the type of structure in the process of interest (e.g., degree of linearity/non-linearity, dependence in the covariates, relative importance of specific covariates, etc.).

Recall in our discussion of boosting and bagging that the notion of "out of bag (OOB) samples" can be quite helpful. In the context of random forests, they are very easy to obtain. Specifically, for each observation pair (x_i, y_i), its random forest predictor is constructed by averaging only the trees corresponding to bootstrap samples in which this observation did **not** appear.

Random Forests (cont.)

Thus, one can compare this OOB prediction to the actual response and get an OOB error that is effectively a cross-validation error estimate. The nice thing about this in the context of random forests is that there is no extra work needed to obtain this. When this error stabilizes, one can assume the model is trained. E.g., consider the example from HTF (2009):

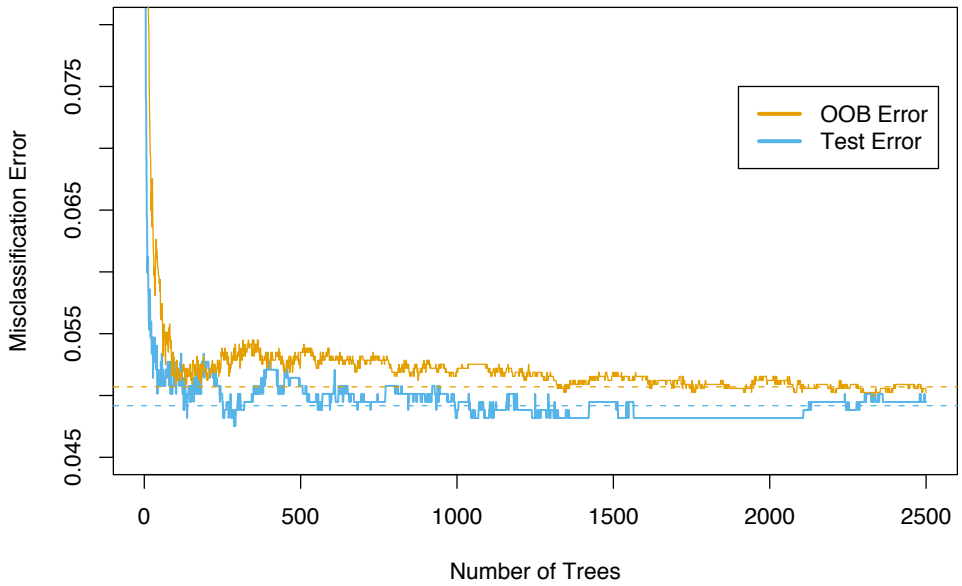

FIGURE 15.4. OOB *error computed on the* `spam` *training data, compared to the test error computed on the test set.*

Random Forests: Variable Importance

CKW

As in CART and Boosting, we consider variable importance as determined by a measure of the improvement in the split-criterion for each variable as aggregated across each tree. Note that random forests typically include more variables because of the random selection, so that it is more likely to see additional "important" variables.

In addition, due to the ease of getting OOB samples, we can construct a different variable importance measure for random forests in terms of the variable's prediction strength. When the bth tree is grown, the OOB samples are passed down the tree and the prediction accuracy is noted. Then, the values for the jth variable are randomly permuted in the OOB samples, and the accuracy is again computed. The decrease in accuracy is averaged over all trees and gives a measure of importance for variable j

Note, this does not measure the effect on prediction if this variable was not available, because the refitted model would likely contain surrogate variables to take its place.

Random Forests: Variable Importance (cont.)

Consider the variable importance plots from HTF (2009):

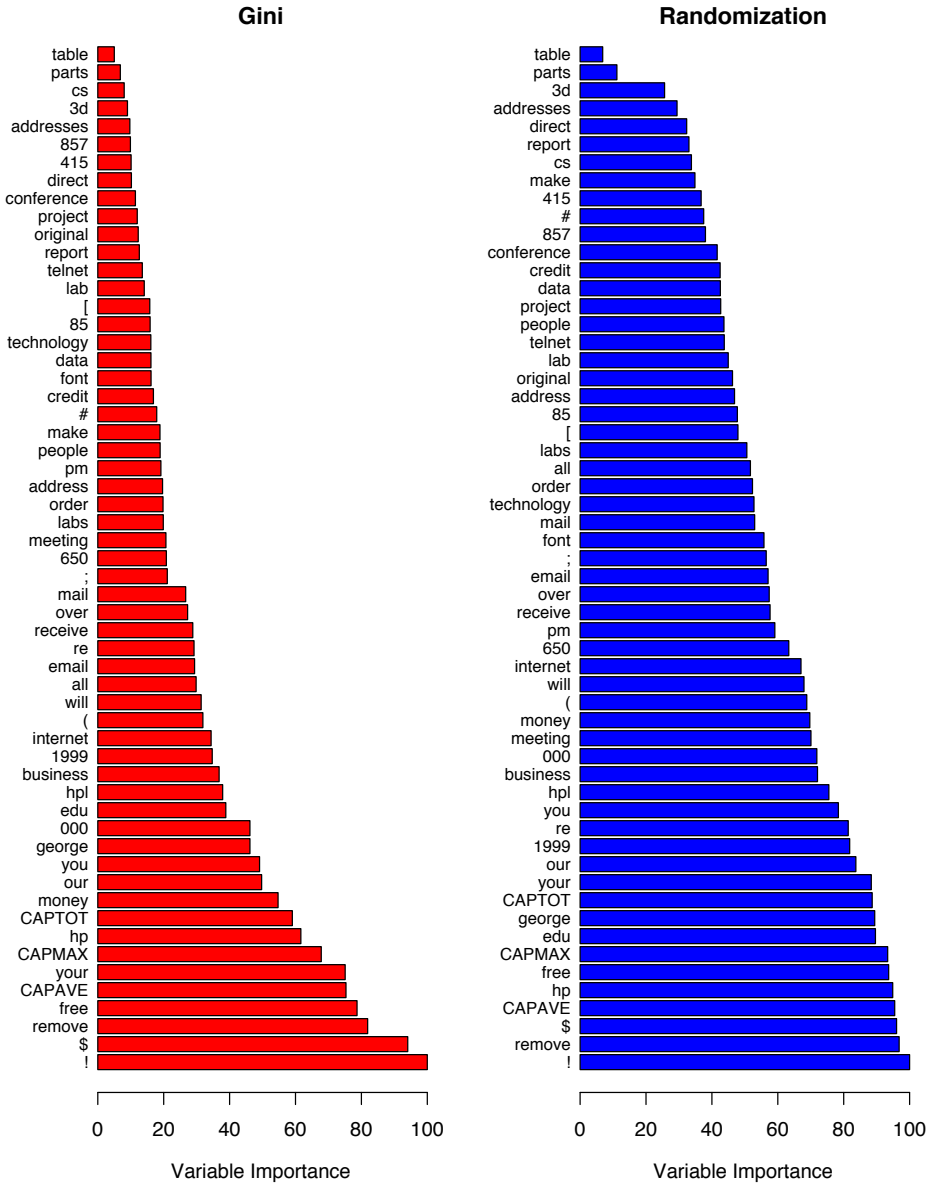

FIGURE 15.5. *Variable importance plots for a classification random forest grown on the* `spam` *data. The left plot bases the importance on the Gini splitting index, as in gradient boosting. The rankings compare well with the rankings produced by gradient boosting (Figure 10.6 on page 316). The right plot uses* OOB *randomization to compute variable importances, and tends to spread the importances more uniformly.*

Random Forests (cont.)

- *Proximity Plots:* Random forest programs often give proximity plots, which give an indication of which observations are effectively close together in terms of the random forest classifier. As discussed in HTF (2009, 15.3.3), these don't often provide much more information than where the points lie relative to the decision boundary.
- *Overfitting:* It is often said that random forests do not over fit. Although they CAN over fit, they are fairly robust to over fitting if the number of relevant variables in $\mathbf{X}$ is reasonably large relative to p. (See HTF (2009, 15.3.4) for more details).
- *Bias:* The bias in random forests, as in bagging, is the same as the bias of the individual tree, so the only improvement in MSE is due to variance reduction. (See HTF (2009, 15.4.2)).
- *Similarity with Nearest Neighbor Classifiers:* It can be shown that the random forest classifier has some similarity to k-nearest neighbor classification methods. (See HTF (2009, 15.4.3)).

Random Forests (cont.)

CKW

- There are several R packages to implement random forests: e.g., `randomForest`, `cforest`, `ParallelForest`, `ranger`, `Rborist`
- You may note a bit of a bias against random forests (compared to gradient boosting) when reading Chapter 15 of HTF (2009). This is natural given that the authors are more intimately connected to gradient boosting. However, keep in mind that both methodologies are very powerful and either one can outperform the other depending on the situation. In my personal experience, random forests have proven to be a very good general nonlinear nonparametric prediction approach when emulating complex nonlinear deterministic models. You may find that you prefer something different; this is the value of having several different learning methodologies available to you, and why it is important to have some familiarity with various software implementations.