Random Forests

Matthew S. Shotwell, Ph.D.

Department of Biostatistics
Vanderbilt University School of Medicine
Nashville, TN, USA

March 25, 2020

Random Forests

- bootstrap aggregation (bagging)
- fit same model to resamples many times
- regression: average predictions
- classification: avarage probs. or “committee vote”
- reduces variance of estimated predictor
- works well for high-var. low-bias predictors (e.g., trees)
- random forests is modified bagging

Random Forests

- resamples are not independent samples
- bagged trees are correlated
- weakens the "wisdom of crowds"
- random forests generates *de-correlated* trees
- performance similar to boosting for many problems

Random Forests

- in bagging, trees are identically distributed (i.d.)
- $\rightarrow$ bias of bagged trees is same as individual tree
- $\rightarrow$ bagging improves variance
- can improve bias and variance using low-bias individual tree
- in boosting, trees are adaptive, not i.d.

Random Forests

- prediction from RF is average of B i.d. predictions
- variance of the average of B i.d. predictions with variance σ^2 and pairwise correlation ρ:

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

- as B grows, second term vanishes
- can only reduce first term by reducing ρ
- random forests: reduce ρ without increasing σ^2 too much

Random Forests

- random forests modifies the tree-growing procedure
- before each split, select $m \leq p$ input variables at random as candidates for splitting
- this reduces correlation among the trees
- typically $m = \lfloor \sqrt{p} \rfloor$ for regression
- typically $m = \lfloor p/3 \rfloor$ for classification
- random forest predictor after B trees grown is

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

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 $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

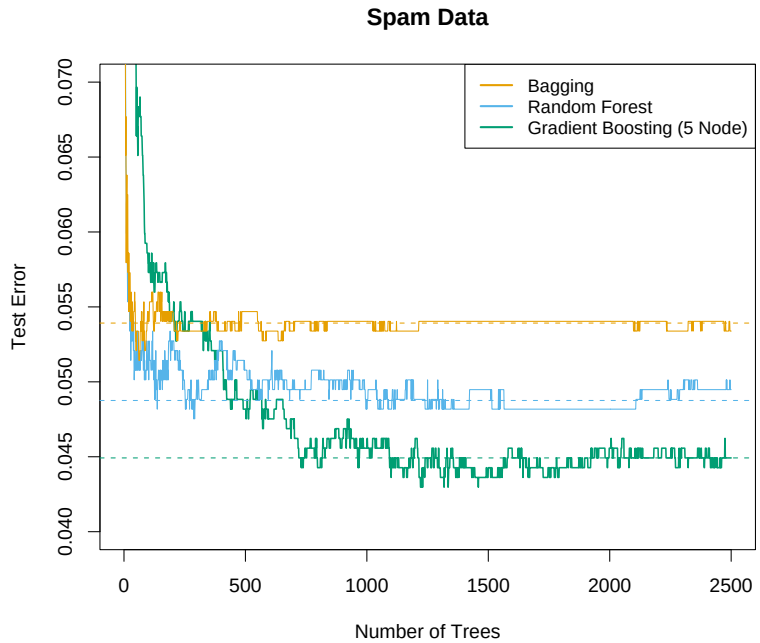


FIGURE 15.1. *Bagging, random forest, and gradient boosting, applied to the spam data. For boosting,* 5*-node trees were used, and the number of trees were chosen by* 10*-fold cross-validation (*2500 *trees). Each "step" in the figure corresponds to a change in a single misclassification (in a test set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

Out-of-bag samples

- random forests uses notion of out-of-bag (OOB) error
- for each $z_i = (x_i, y_i)$ construct its random forest predictor by averaging only those trees corresponding to bootstrap samples in which z_i was not sampled
- OOB error like LOO bootstrap validation, or N-fold CV error
- trees are added until OOB error "stabilizes"

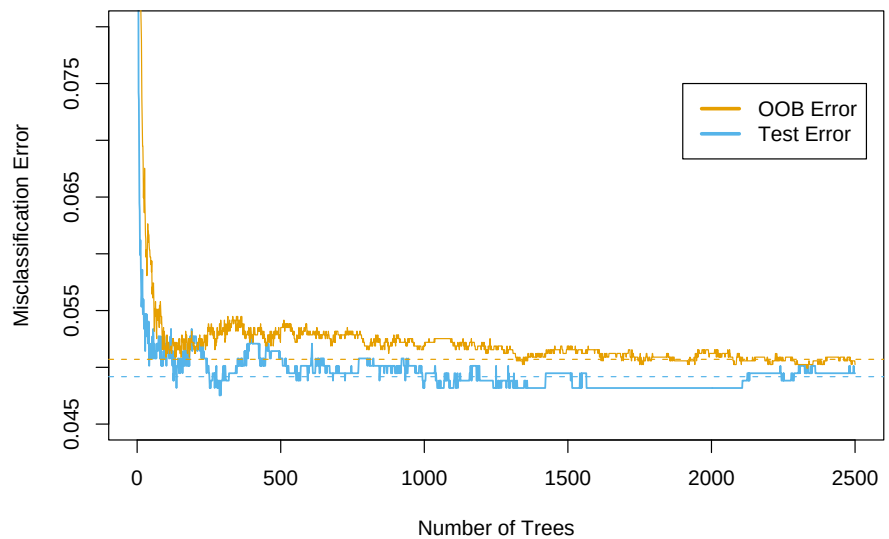


FIGURE 15.4. OOB *error computed on the* `spam` *training data, compared to the test error computed on the test set.*

Overfitting and tuning parameters

- random split-candidate selection increases chance that noise variable will be used for splitting
- when many noise variables, random forest may overfit
- need to modify number of candidates $m \leq p$
- unlike best-subset, boosting, number of trees B in random forest (and other bagged predictors) not as sensitve to overfitting

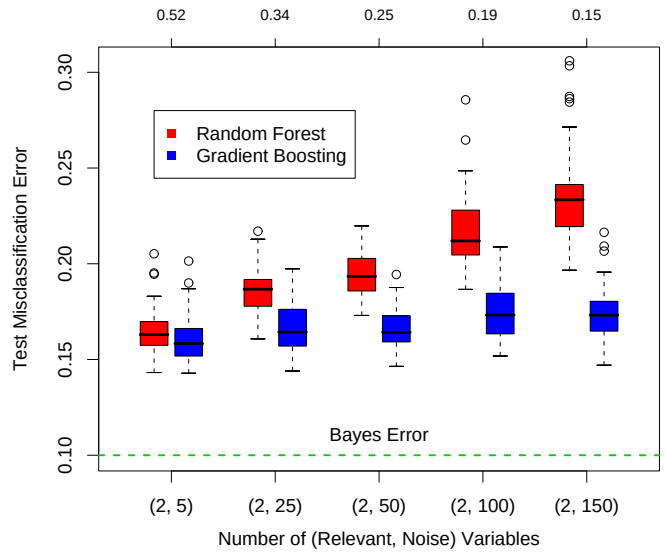


FIGURE 15.7. *A comparison of random forests and gradient boosting on problems with increasing numbers of noise variables. In each case the true decision boundary depends on two variables, and an increasing number of noise variables are included. Random forests uses its default value* $m = \sqrt{p}$. *At the top of each pair is the probability that one of the relevant variables is chosen at any split. The results are based on* 50 *simulations for each pair, with a training sample of* 300, *and a test sample of* 500.

Overfitting and tuning parameters

Random forest tuning parameters:

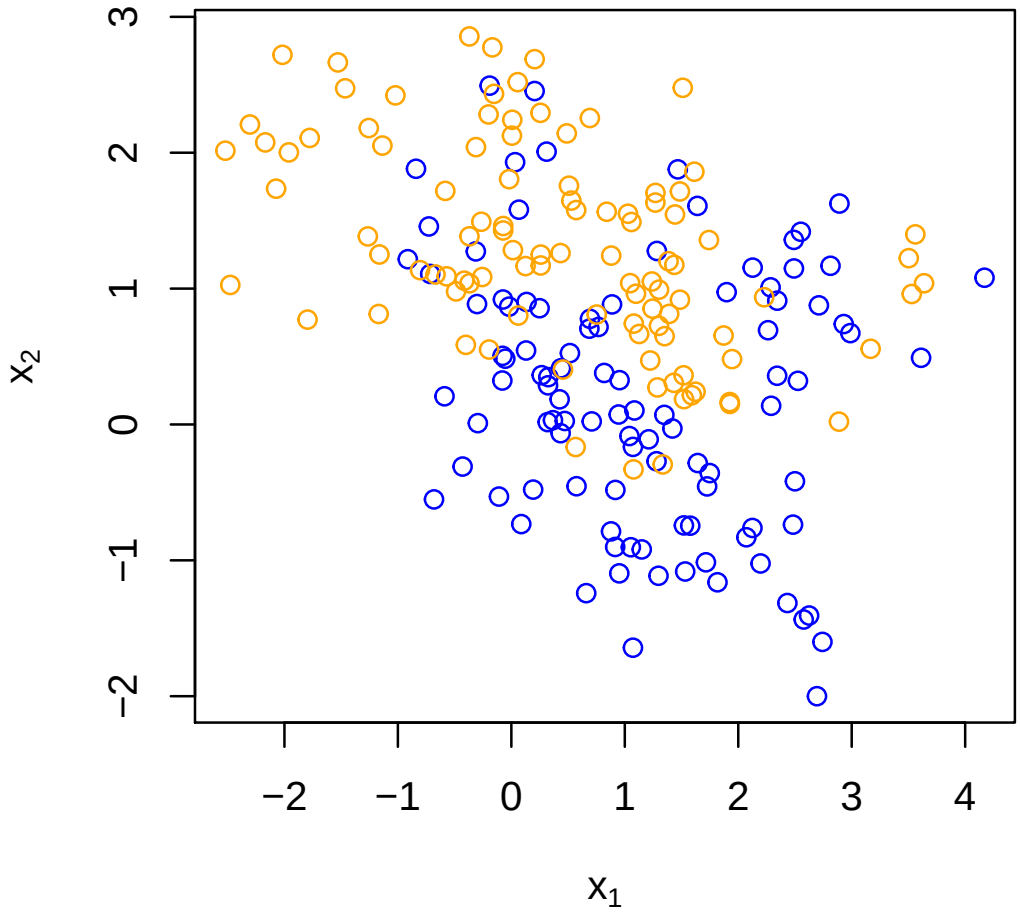
- B - number of trees in forest
- m - number of features considered at each split
- maximum tree depth
- minimum required data points in terminal node

Compute CV-error or OOB error on grid of tuning parameter values, then select the values with smallest error

Variable importance: I^2

- variable importance idea: for variable X, add up improvement (reduced training error) associated with each split on a variable X, within and across trees

Variable importance: I^2



Variable importance: I^2

- for single tree T, "squared relevance" of variable l is

$$I_l^2(T) = \sum_{t=1}^{J-1} \hat{i}_t^2 I(v(t) = l)$$

- t index the $J-1$ internal nodes
- $v(t)$ is split variable at node t
- $\hat{i}_t^2$ is squared improvement in trainin error by splitting
- for tree ensembles trees, average over all B trees

$$I_l^2 = \frac{1}{B} \sum_{b=1}^{B} I_l^2(T_b)$$

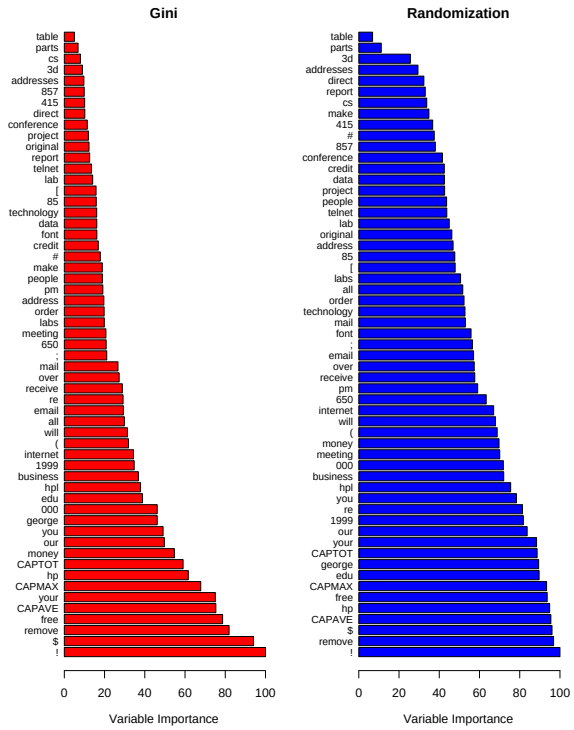


FIGURE 15.5. *Variable importance plots for a classification random forest grown on the* `spam` *data. The left plot bases the importance on the Gini splitting index, as in gradient boosting. The rankings compare well with the rankings produced by gradient boosting (Figure 10.6 on page 316). The right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

Variable importance: OOB randomization

- prediction accuracy of b^{th} tree computed using OOB error
- values for j^{th} input variable permuted (voids contribution of variable)
- prediction accuracy recomputed after permutation
- decrease in accuracy averaged across all trees
- not same as model without permuted variable

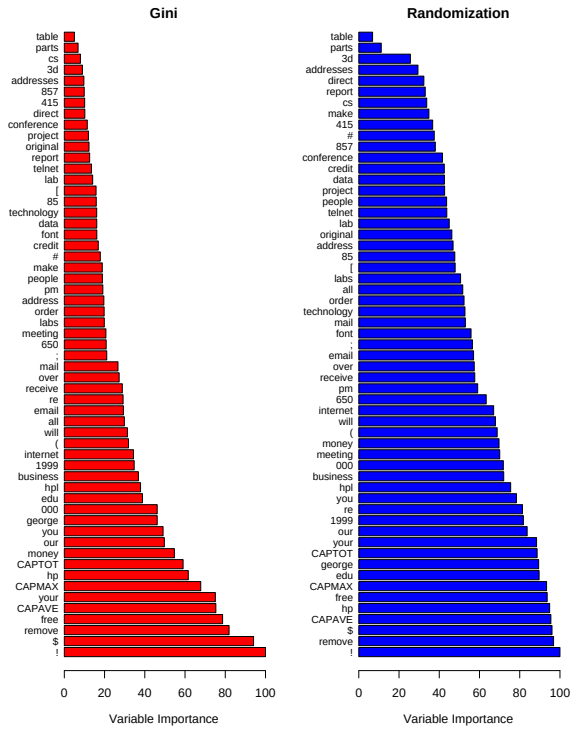


FIGURE 15.5. *Variable importance plots for a classification random forest grown on the* `spam` *data. The left plot bases the importance on the Gini splitting index, as in gradient boosting. The rankings compare well with the rankings produced by gradient boosting (Figure 10.6 on page 316). The right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

Proximity plots

- for every pair of training points, proximity is the number of trees in which the pair shares a terminal node (gives a proximity/similarity matrix)
- multidimensional scaling (MDS) finds a q-dimensional representation of the training data features (p-dimensional) such that the euclidean distance between pairs is good estimate of proximity between pairs
- MDS used to generate low-dimension representation of data that preserves proximity (e.g., q=2)
- proximity plots generally have star shape, one arm per class
- "pure" regions tend to map to extremity of star
- "impure" regions tend to map to center of star

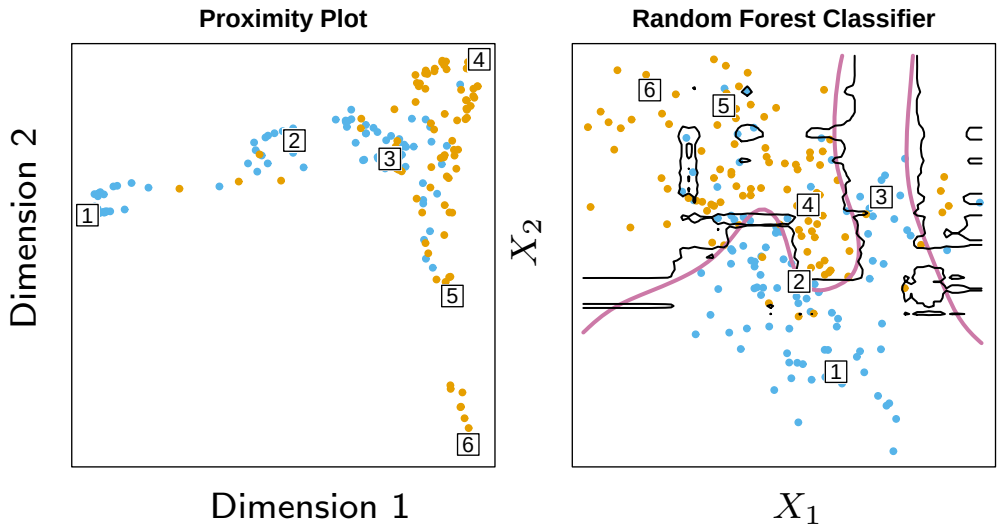


FIGURE 15.6. *(Left): Proximity plot for a random forest classifier grown to the mixture data. (Right): Decision boundary and training data for random forest on mixture data. Six points have been identified in each plot.*

Random Forests

- R package `randomForest`