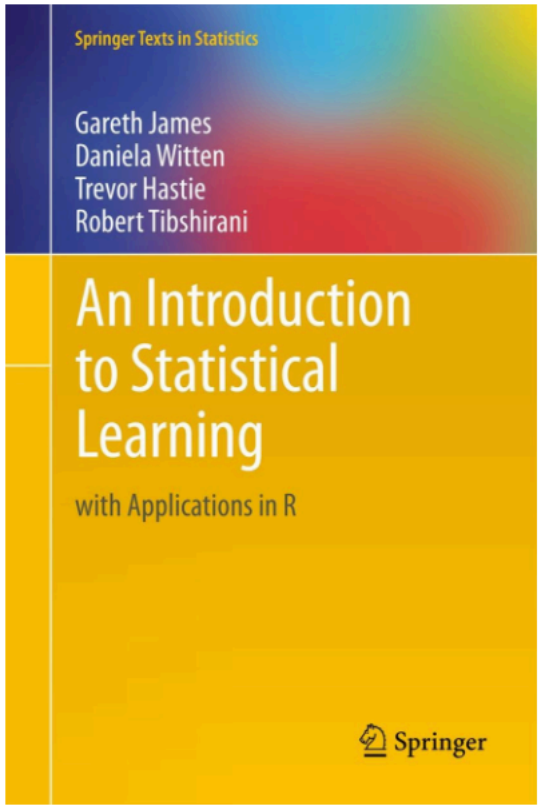
9. Ensemble Methods

ESC Spring 2018 – Data Mining and Analysis

SeoHyeong Jeong



Textbook:

An Introduction to Statistical Learning

Lecture Slides:

Stanford Stats 202: Data Mining and Analysis

Spring 17’ ESC Statistical Data Analysis

Reading:

An Introduction to Statistical Learning

chapter 8.2 Bagging, Random Forests, Boosting

Table of Contents

Decision Trees, Summary

- Grow the tree by recursively splitting the samples in the leaf R_i according to $X_j > s$, such that (R_i, X_j, s) maximize the drop in RSS.
  -> Greedy algorithm.

- Create a sequence of subtrees $T_0, \dots, T_m$ using a pruning algorithm.

- Select the best tree T_i (or the best α) by cross validation.
  -> Why might it be better to choose α instead of the tree T_i by cross-validation?

Bagging

- Bagging = Boostrap Aggregating
- In the Bootstrap, we replicate our dataset by sampling with replacement:
  - Original dataset: $x = c(x1, x2, \dots, x100)$
  - Boostrap samples:

$$boot1 = sample(x, 100, replace = True), \dots,$$
$$bootB = sample(x, 100, replace = True).$$

- We used these samples to get the Standard Error of a parameter estimate:

$$\mathrm{SE}(\hat{\beta}_1) \approx \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\hat{\beta}_1^{(b)} - \frac{1}{B}\sum_{k=1}^{B}\hat{\beta}_1^{(k)})^2}$$

- The decision trees suffer from high variance.
- Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical model.
- In Bagging we average the predictions of a model fit to many Bootstrap samples. The predicted model is obtain as:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

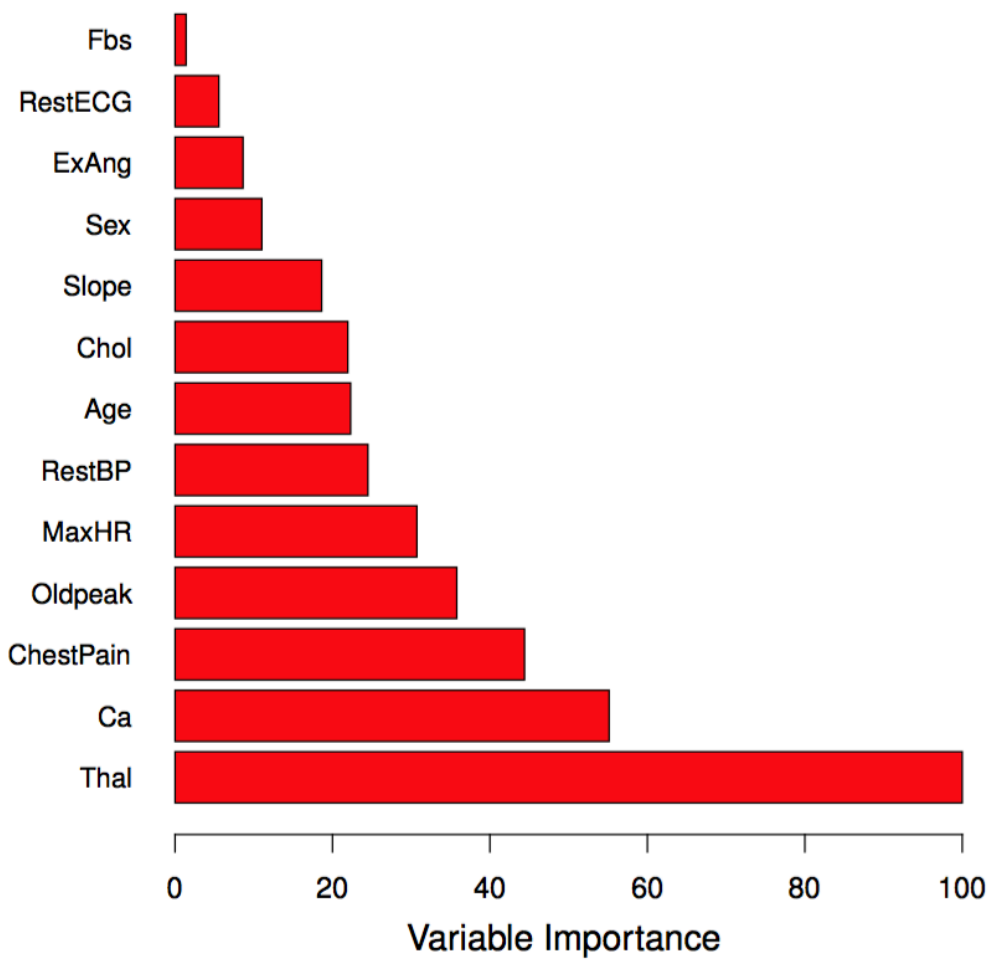
When Does Bagging Make Sense?

- When a regression method or a classifier has a tendency to overfit, Bagging reduces the variance of the prediction.
  - When n is large, the empirical distribution is similar to the true distribution of the samples.
  - Bootstrap samples are like independent realizations of the data.
  - Bagging amounts to averaging the fits from many independent datasets, which would reduce the variance by a factor $1/B$. Where B is the number of how many samples were bootstrapped.

Bagging Decision Trees

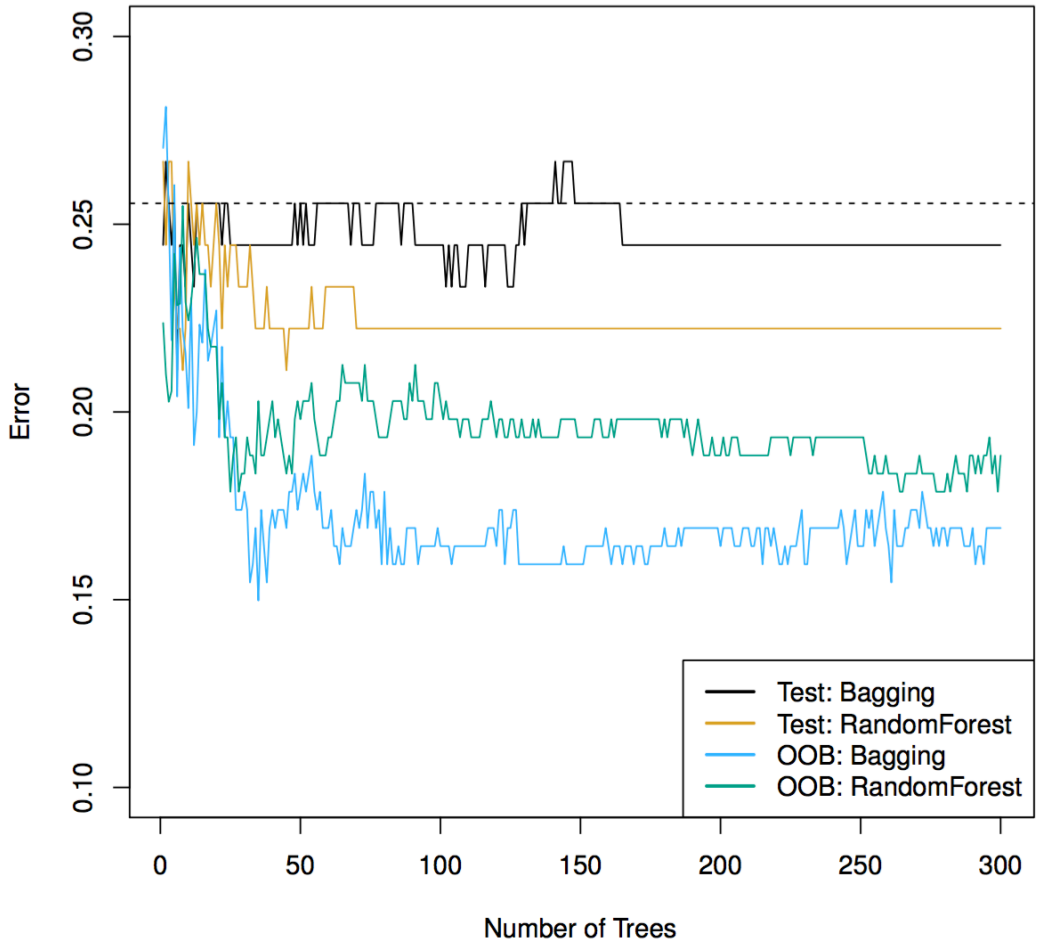
- Disadvantage: Every time we fit a decision tree to a Bootstrap sample, we get a different tree T^b.

  -> Loss of interpretability
- For each predictor, add up the total amount by which the RSS (or Gini index) decreases every time we use the predictor in T^b.
- Average this total over each Bootstrap estimate $T^1, \dots, T^B$.



Out -Of-Bag (OOB) Error

- To estimate the test error of a bagging estimate, we could use cross-validation.
- Each time we draw a Bootstrap sample, we only use 63% of the observations.
- **Idea:** use the rest of the observations as a test set.
- **OOB error:**
  - For each sample x_i, find the prediction $\hat{y}_i^b$ for all bootstrap samples b which do not contain x_i. Average these predictions to obtain $\hat{y}_i^{oob}$.
  - Compute the error $(y_i - \hat{y}_i^{oob})^2$.
  - Average the errors over all observations $i = 1, \dots, n$.



The test error decreases as we increase B

(dashed line is the error for a plain decision tree).

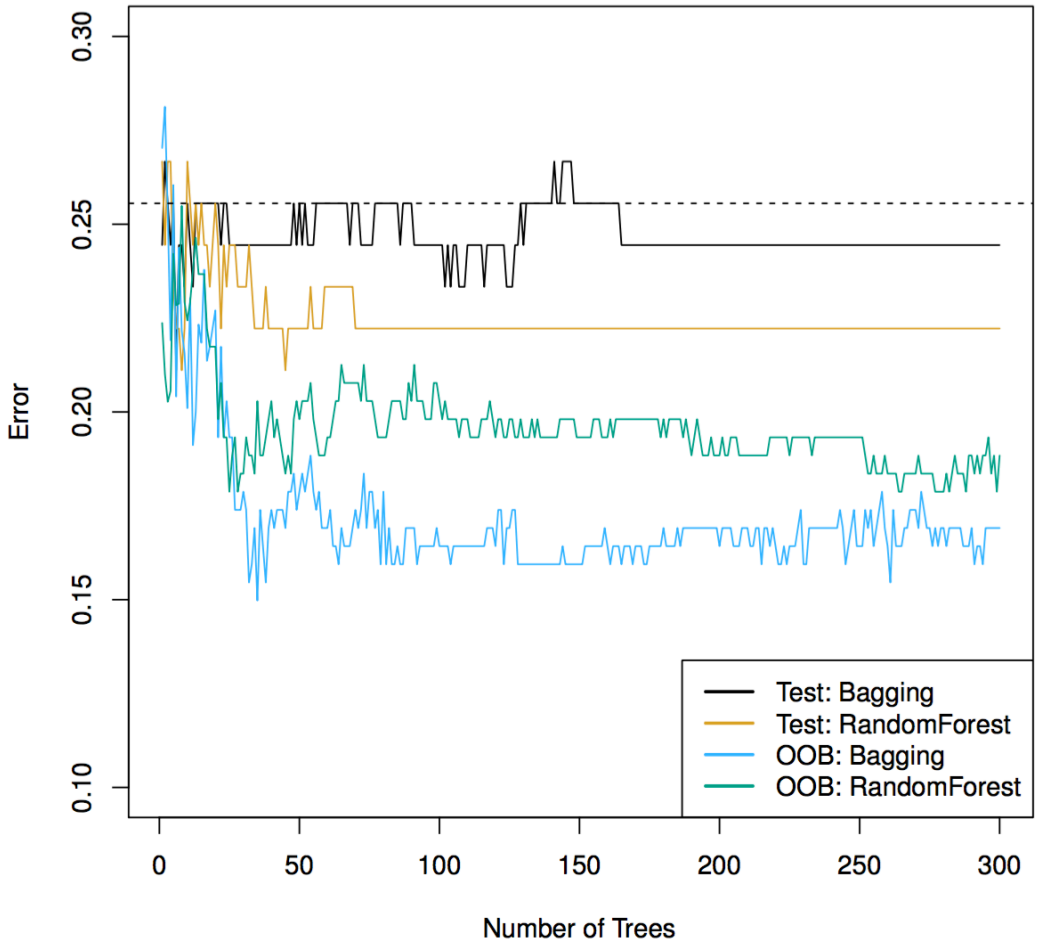
Random Forests

- Bagging has a problem:

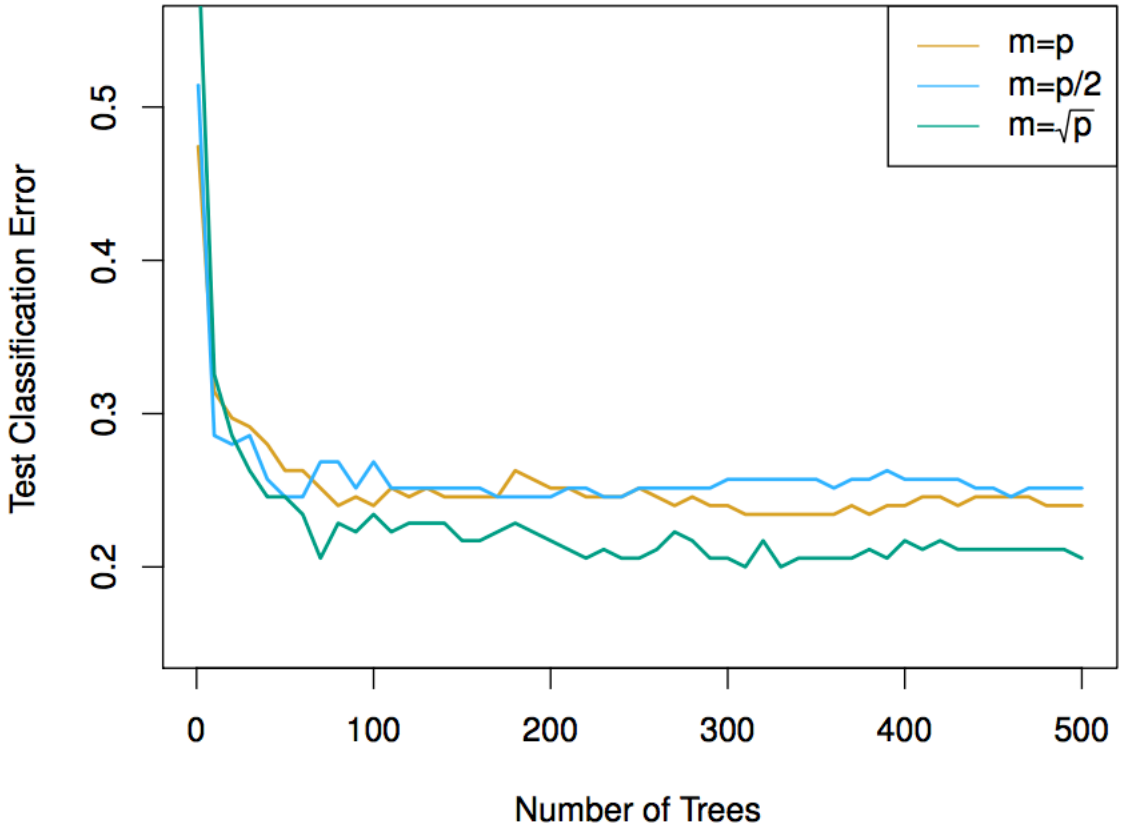
  -> The trees produced by different Bootstrap samples can be very similar.

- **Random Forest:**
  - We fit a decision tree to different Bootstrap samples.
  - When growing the tree, we select a random sample of $m < p$ predictors to consider in each step.
  - This will lead to very different (or "uncorrelated") trees from each sample.
  - Finally, average the prediction of each tree.

Random Forests vs. Bagging



Random Forest, Choosing m



The optimal m is usually around $\sqrt{p}$,

but this can be used as a tuning parameter.

Boosting

- In Boosting, the trees are grown **sequentially**: each tree is grown using information from previously grown trees.
- **Boosting learns slowly**: we typically fit small (low variance but high bias) trees in each round.
- Boosting trains on errors than on the outcome Y.
- **You can boost anything**: you can apply boosting to any predictive learning method, not just decision trees!

Boosting Algorithm

- Boosting has three tuning parameters:

  1. The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly it at all. We use cross-validation to select B.
  2. The shrinkage parameter λ, a small positive number. This controls the rate at which boosting learns. Very small λ can require using a very large value of B in order to achieve good performance.
  3. The number d of splits in each tree, which controls the complexity f the boosted ensemble.

Algorithm 8.2 *Boosting for Regression Trees*

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.

2. For $b = 1, 2, \ldots, B$, repeat:

   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).

   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$

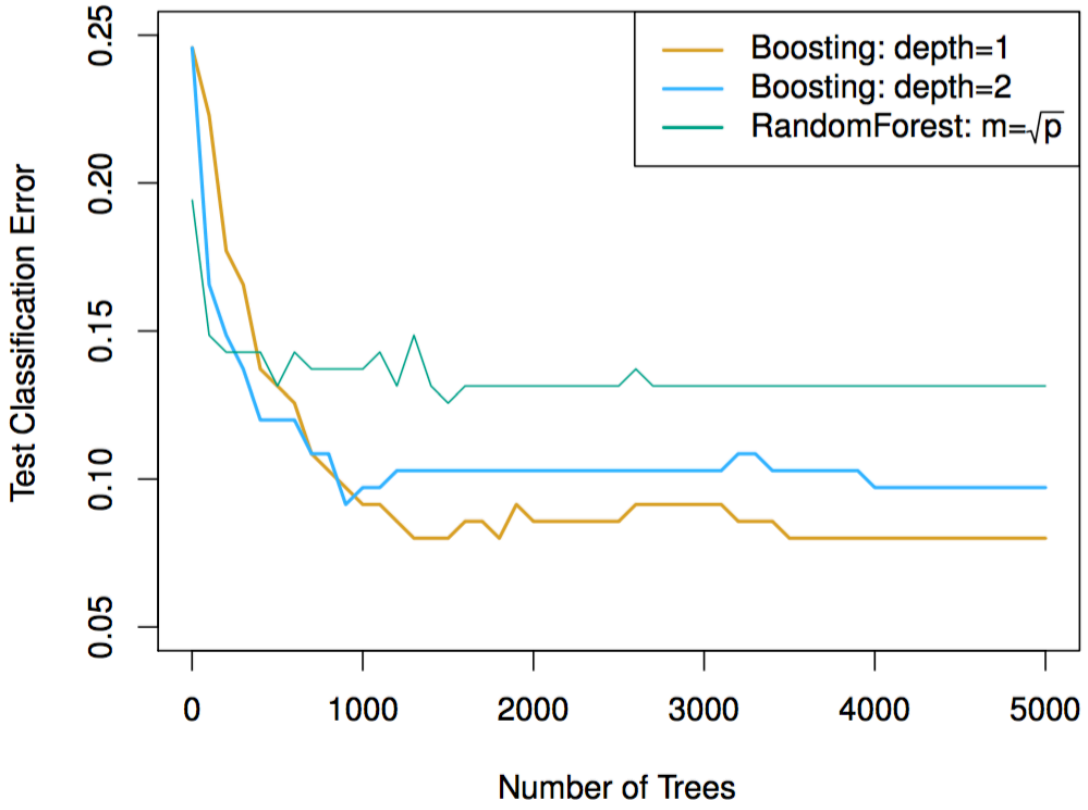
   (c) Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

Boosting vs. Random Forests



The parameter $\lambda = 0.01$ in each case.

We can tune the model by CV using λ, d, B.