

Lecture 20: Bagging, Random Forests, Boosting

Reading: Chapter 8

STATS 202: Data mining and analysis

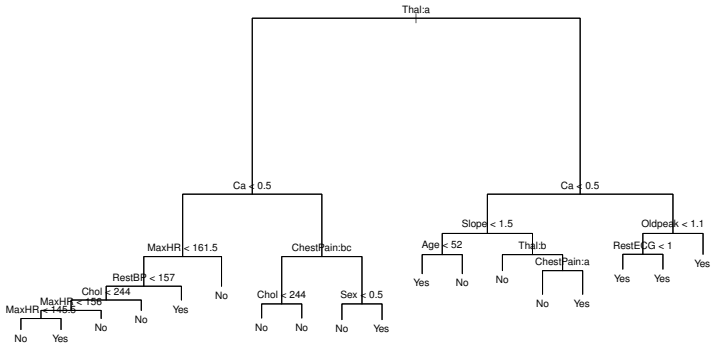
November 8, 2019

Classification and Regression trees, in a nut shell

- ▶ 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, T_1, \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?

Example. Heart dataset.

How do we deal with categorical predictors?



Categorical predictors

- ▶ If there are only 2 categories, then the split is obvious. We don't have to choose the splitting point s , as for a numerical variable.
- ▶ If there are more than 2 categories:
 - ▶ Order the categories according to the average of the response:

ChestPain : a > ChestPain : c > ChestPain : b

- ▶ Treat as a numerical variable with this ordering, and choose a splitting point s .
- ▶ One can show that this is the optimal way of partitioning.

Missing data

- ▶ Suppose we can assign every sample to a leaf R_i despite the missing data.
- ▶ When choosing a new split with variable X_j (growing the tree):
 - ▶ Only consider the samples which have the variable X_j .
 - ▶ In addition to choosing the best split, choose a second best split using a different variable, and a third best, ...
- ▶ To propagate a sample down the tree, if it is missing a variable to make a decision, try the second best decision, or the third best, ...

Bagging

- ▶ Bagging = Bootstrap Aggregating
- ▶ In the Bootstrap, we replicate our dataset by sampling with replacement:
 - ▶ Original dataset: $x = c(x_1, x_2, \dots, x_{100})$
 - ▶ Bootstrap samples:
`boot1 = sample(x, 100, replace = True), ...`
`bootB = sample(x, 100, replace = True).`
- ▶ We used these samples to get the Standard Error of a parameter estimate:

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

Bagging

- ▶ In **Bagging** we average the predictions of a model fit to many Bootstrap samples.

Example. Bagging the Lasso

- ▶ Let $\hat{y}^{L,b}$ be the prediction of the Lasso applied to the b th bootstrap sample.
- ▶ Bagging prediction:

$$\hat{y}^{\text{boot}} = \frac{1}{B} \sum_{b=1}^B \hat{y}^{L,b}.$$

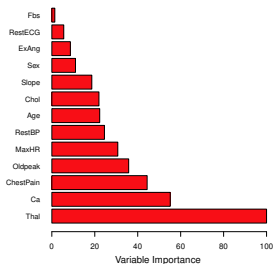
When does Bagging make sense?

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 $1/B$.

Bagging decision trees

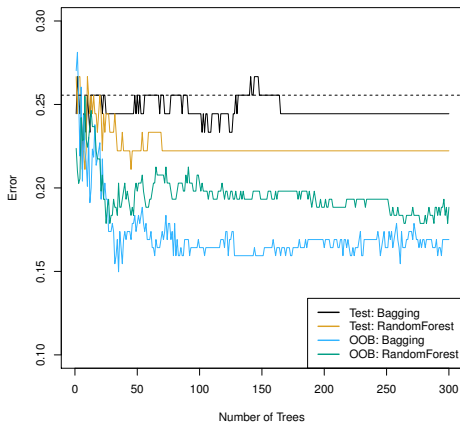
- ▶ **Disadvantage:** Every time we fit a decision tree to a Bootstrap sample, we get a different tree T^b .
→ Loss of interpretability
- ▶ For each predictor, add up the total amount by which the RSS (or Gini index) decreases every time we use the predictor in T^b .
- ▶ Average this total over each 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 . There should be around $0.37B$ of them. Average these predictions to obtain \hat{y}_i^{oob} .
 - ▶ Compute the error $(y_i - \hat{y}_i^{\text{oob}})^2$.
 - ▶ Average the errors over all observations $i = 1, \dots, n$.

Out-of-bag (OOB) error



The test error decreases as we increase B
(dashed line is the error for a plain decision tree).

Random Forests

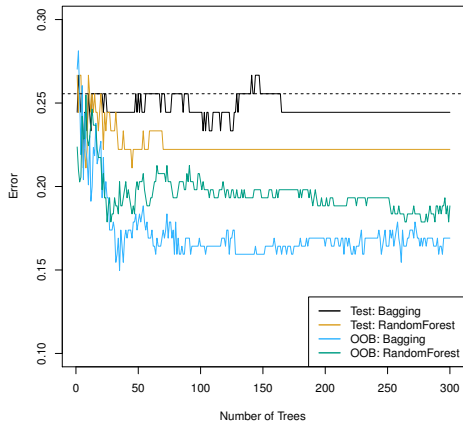
Bagging has a problem:

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

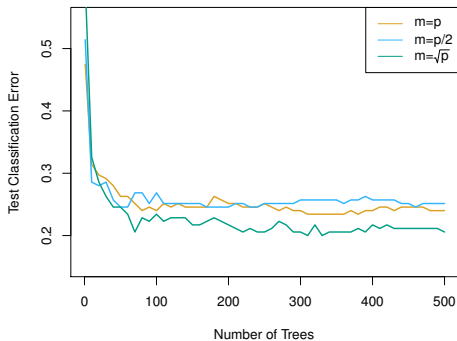
Random Forests:

- ▶ 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 Forests, choosing m



The optimal m is usually around \sqrt{p} ,
but this can be used as a tuning parameter.

Boosting

1. Set $\hat{f}(x) = 0$, and $r_i = y_i$ for $i = 1, \dots, n$.
2. For $b = 1, \dots, B$, iterate:
 - 2.1 Fit a decision tree \hat{f}^b with d splits to the response r_1, \dots, r_n .
 - 2.2 Update the prediction to:

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

- 2.3 Update the residuals,

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

3. Output the final model:

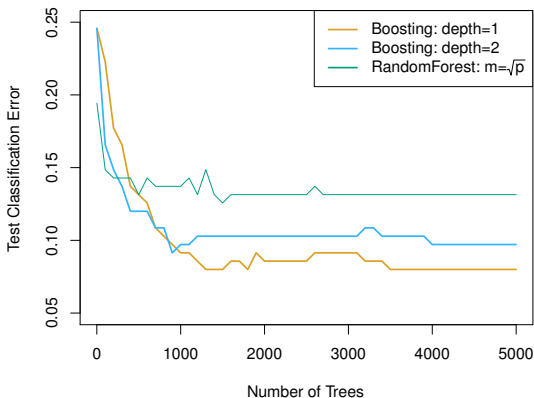
$$\hat{f}(x) = \sum_{b=1}^B \lambda \hat{f}^b(x).$$

Boosting, intuitively

Boosting learns *slowly*:

We first use the samples that are easiest to predict, then slowly down weigh these cases, moving on to harder samples.

Boosting vs. random forests



The parameter $\lambda = 0.01$ in each case.
We can tune the model by CV using λ, d, B .