

Lecture 21: Bagging, Random Forests, Boosting

Reading: Chapter 8

STATS 202: Data mining and analysis

Lester Mackey
November 9, 2015

(Slide credits: Sergio Bacallado)

Review: Categorical predictors in decision trees

- ▶ If there are only 2 categories, then the two branches of the tree correspond to the two categories.
- ▶ If there are more than 2 categories, need to divide categories into two groups in a way that minimizes training error. There is a shortcut for regression or binary classification trees:
 - ▶ Order the categories according to the average of the response:

ChestPain : a > ChestPain : c > ChestPain : b

- ▶ Order by average output value y_i in regression case
 - ▶ Order by proportion of $y_i = 1$ in $\{0, 1\}$ classification case
- ▶ Choose a splitting point in this ordering.

Bagging

- ▶ Bagging = Bootstrap Aggregating
 - ▶ A new application of the bootstrap!
- ▶ 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 often use these samples to approximate the Standard Error of a parameter estimate:

$$SE(\hat{\beta}_1) \approx SD(\hat{\beta}_1^{(1)}, \dots, \hat{\beta}_1^{(B)})$$

Bagging = Bootstrap Aggregation

- ▶ In **Bagging** we average the *predictions* of a model fit to many Bootstrap samples.
 - ▶ You can bag any predictive model!
 - ▶ *Example.* Bagging the Lasso
 - ▶ Let $\hat{y}^{L,b}$ be the prediction of the Lasso applied to the b th bootstrap sample.
 - ▶ Bagging prediction:

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

Bagging = Bootstrap Aggregation

- ▶ Replicate the dataset by sampling with replacement.
- ▶ We apply a learning method to each bootstrap replicate, to produce predictions $\hat{f}^{(1)}, \dots, \hat{f}^{(B)}$.
- ▶ In Chapter 5, we were interested in the variability of these predictions:

$$\text{SE}(\hat{f}(x)) \approx \text{SD}(\hat{f}^{(1)}(x), \dots, \hat{f}^{(B)}(x)).$$

- ▶ Now, we will use the average of these predictions as an estimator with reduced variance:

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

When does Bagging make sense?

Bagging serves to reduce the variance of a prediction

- ▶ Especially useful when combined with higher-variance / more flexible prediction methods

Roughly, 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 is akin to averaging the fits from many independent datasets, which would reduce the variance by a factor $1/B$
- ▶ This is all just an approximation to the truth!

Bagging decision trees

- ▶ Replicate the dataset by sampling with replacement.
- ▶ Fit a decision tree to each bootstrap replicate (growing the tree and pruning).
- ▶ **Regression:** To make a prediction for an input point x , average the predictions of all the trees:

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

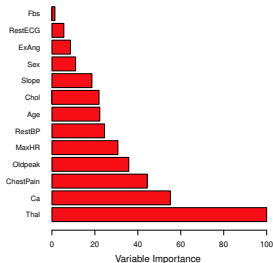
- ▶ **Classification:** To make a prediction for an input point x_0 , there are several reasonable options:
 - ▶ Take the majority vote from the set of predictions

$$\hat{y}_0^{(1)}, \dots, \hat{y}_0^{(B)}.$$

- ▶ Average the class label proportions from each replicate tree and predict the label with the highest average proportion

Bagging decision trees

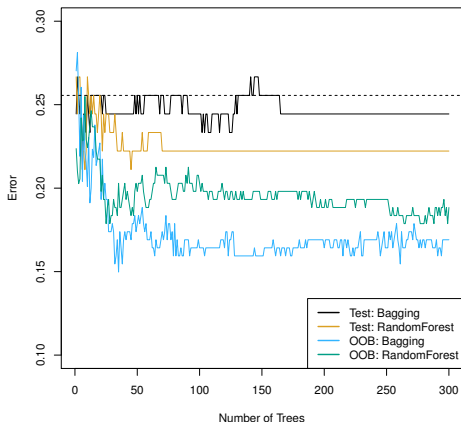
- ▶ **Disadvantage:** Every time we fit a decision tree to a Bootstrap sample, we get a different tree T^b .
 - Loss of interpretability
- ▶ However, can compute a measure of variable importance
 - ▶ For each predictor, add up the total amount by which the RSS (or Gini index) decreases every time we 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.
 - ▶ But this might be computationally expensive
- ▶ Recall that 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 $\hat{y}_i^{\text{oob}}$.
 - ▶ Example on board
 - ▶ Compute the error $(y_i - \hat{y}_i^{\text{oob}})^2$.
 - ▶ Average the errors over all observations $i = 1, \dots, n$.
- ▶ For B large, OOB error is virtually equivalent to LOOCV.

Out-of-bag (OOB) error



- ▶ The test error converges as we increase number of trees B (dashed line is the error for a single classification tree).
- ▶ Erratum: OOB lines for bagging and RF are swapped!

Random Forests

Bagging has a weakness:

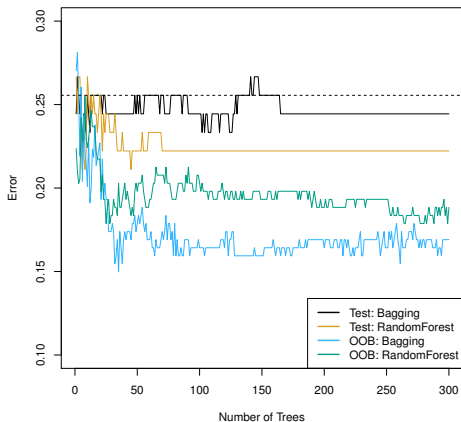
→ The trees produced by different Bootstrap samples can be very similar.

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 at each branch of the tree.
- ▶ This will lead to less similar trees with less correlated predictions.
- ▶ Finally, average the prediction of each tree.

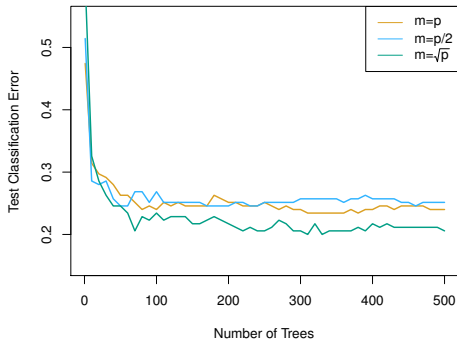
(Related method: Bayesian Additive Regression Trees)

Random Forests vs. Bagging



- ▶ Erratum: OOB lines for bagging and RF are swapped!
- ▶ Random forests trained with $m = \sqrt{p}$
- ▶ Bagging trees = Random forests with $m = p$

Random Forests, choosing m



- ▶ Choosing $m = \sqrt{p}$ is often a reasonable default, but this can be used as a tuning parameter.
- ▶ As in bagging, test error will converge / stabilize as number of trees B gets large, so typically choose B large enough for error estimate to have stabilized

Boosting

Sequential procedure for combining (“ensembling”) decision trees

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

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

- 3.3 Update the residuals,

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

4. Output the final model:

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

Properties of boosting

Boosting learns sequentially, trains on errors

Each iteration learns to predict the mistakes of the previous iteration.

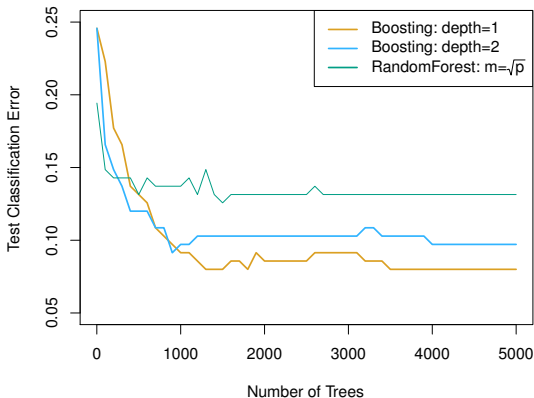
Boosting learns *slowly*

We typically fit small (lower variance but higher bias) trees on each round.

You can boost anything

You can apply boosting to any predictive learning method, not just decision trees!

Boosting vs. random forests



- ▶ The parameter $\lambda = 0.01$ in each case.
- ▶ We can tune the model by CV using λ , d , and $\#$ of trees B .
- ▶ A single-split decision tree ($d = 1$) is called a **decision stump**