Lecture 21: Bagging, Random Forests, Boosting

Reading: Chapter 8

STATS 202: Data mining and analysis

Sergio Bacallado

November 10, 2014

Bagging = Bootstrap Aggregation

Bagging = Bootstrap Aggregation

- Replicate the dataset by sampling with replacement.

Bagging = Bootstrap Aggregation

- Replicate the dataset by sampling with replacement.
- We apply a learning method to each bootstrap replicate, to produce predictions $\hat{f}^{(1)}, \ldots, \hat{f}^{(B)}$.

Bagging = Bootstrap Aggregation

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

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

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:

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

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

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

Bagging decision trees

- Replicate the dataset by sampling with replacement.

Bagging decision trees

- Replicate the dataset by sampling with replacement.
- Fit a decision tree to each bootstrap replicate (growing the tree, and pruning).

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:

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

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, take the majority vote from the set of predictions:

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

Bagging decision trees

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

Bagging decision trees

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

  $\rightarrow$ Loss of interpretability

Bagging decision trees

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

  $\rightarrow$ 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.

Bagging decision trees

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

  $\rightarrow$ Loss of interpretability

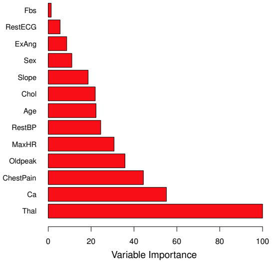
- For each predictor, add up the total amount by which the RSS (or Gini index) decreases every time we use the predictor in T^b.
- Average this total over each Boostrap estimate $T^1, \ldots, T^B$.

Bagging decision trees

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

  $\rightarrow$ 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 Boostrap estimate $T^1, \ldots, T^B$.



Out-of-bag (OOB) error

- To estimate the test error of a bagging estimate, we could use cross-validation.

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.

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.

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 $\hat{y}_i^{\text{oob}}$.

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 $\hat{y}_i^{\text{oob}}$.
  - Compute the error $(y_i - \hat{y}_i^{\text{oob}})^2$.

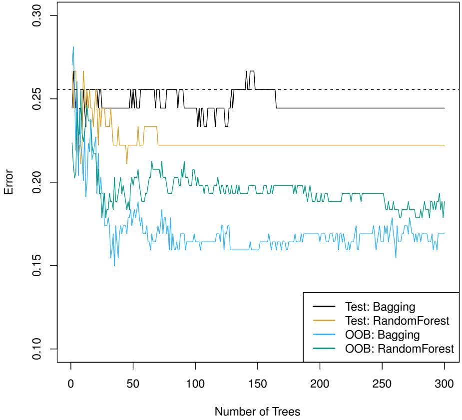
Out-of-bag (OOB) error

- 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 $\hat{y}_i^{\text{oob}}$.
  - Compute the error $(y_i - \hat{y}_i^{\text{oob}})^2$.
  - Average the errors over all observations $i = 1, \ldots, n$.

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 $\hat{y}_i^{\text{oob}}$.
  - Compute the error $(y_i - \hat{y}_i^{\text{oob}})^2$.
  - Average the errors over all observations $i = 1, \ldots, n$.
- For B large, OOB error is virtually equivalent to LOOCV.

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:

$\rightarrow$ The trees produced by different Bootstrap samples can be very similar.

Random Forests:

Random Forests

Bagging has a problem:

$\rightarrow$ The trees produced by different Bootstrap samples can be very similar.

Random Forests:

- We fit a decision tree to different Bootstrap samples.

Random Forests

Bagging has a problem:

$\rightarrow$ 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.

Random Forests

Bagging has a problem:

$\rightarrow$ 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.

Random Forests

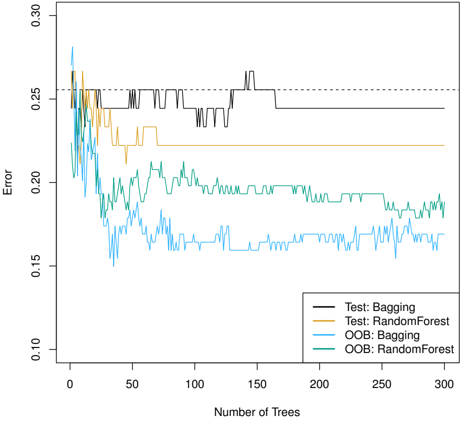
Bagging has a problem:

$\rightarrow$ The trees produced by different Bootstrap samples can be very similar.

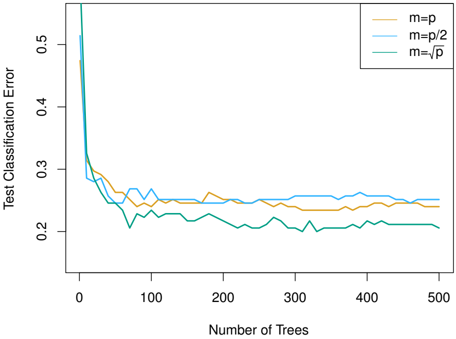
Random Forests:

- 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

Boosting

1. Set $\hat{f}(x) = 0$, and $r_i = y_i$ for $i = 1, \ldots, n$.

Boosting

1. Set $\hat{f}(x) = 0$, and $r_i = y_i$ for $i = 1, \ldots, n$.
2. For $b = 1, \ldots, B$, iterate:

Boosting

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

Boosting

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

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

Boosting

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

Boosting

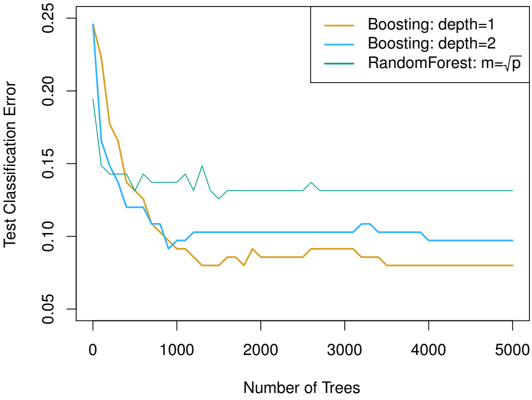
1. Set $\hat{f}(x) = 0$, and $r_i = y_i$ for $i = 1, \ldots, n$.
2. For $b = 1, \ldots, B$, iterate:
   2.1 Fit a decision tree $\hat{f}^b$ with d splits to the response $r_1, \ldots, r_n$.
   2.2 Update the prediction to:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$
   2.3 Update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