STA 314: Statistical Methods for Machine Learning I

Lecture 11 - Bagging, random forest and boosting; Bootstrap

Xin Bing

Department of Statistical Sciences
University of Toronto

- Homework 4 is due tonight.
- The last tutorial is on Dec 4th.
- Please fill out the course evaluation form! Your feedbacks are extremely helpful for future improvement of this course!

Review on simple decision trees

- Decision trees
  - Regression tree
  - Classification tree
- Self-explanatory
- Recursive binary splitting
- Could be deeply grown trees, with large variance
- Pruning

Bootstrap

- Bootstrap is a widely used **resampling** (of the available data) approach!
- It can be used to assess the uncertainty of basically any statistical procedure.
  - For instance, it can be used to estimate the standard errors of the estimated coefficients of a linear model.
  - Much more generally, it can even estimate the whole distribution of the estimated coefficients of a linear model.
- Its validity is backed up by a very general theory!

A simple example

- Suppose that we wish to invest 10K dollars in two financial assets that yield returns of X and Y, respectively, where X and Y are random quantities.
- For any $\alpha \in [0, 1]$, we will invest a fraction α of our money in X, and will invest the remaining $(1 - \alpha)$ in Y.

$$\Big[\alpha X + (1 - \alpha) Y\Big] \times 10{,}000$$

- We wish to choose α to minimize the total risk, or variance, of our investment, that is,

$$\min_{\alpha \in [0,1]} \mathrm{Var}(\alpha X + (1 - \alpha) Y).$$

Example

- One can show that the value of α that minimizes the risk is given by

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}},$$

where $\sigma_Y^2 = \text{Var}(Y)$, $\sigma_X^2 = \text{Var}(X)$ and $\sigma_{XY} = \mathrm{Cov}(X, Y)$.

- If we have past observations $(x_1, y_1), ..., (x_{100}, y_{100})$, we can estimate α by

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}.$$

- How to estimate the variance of the estimator $\hat{\alpha}$?

An Oracle Approach

If we know the distribution of X and Y (usually unknown in reality), we can estimate the variance of the estimator $\hat{\alpha}$ by the following strategy.

- We simulate 100 paired observations of X and Y and compute

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}.$$

- We repeat this procedure 1000 times, and get $\hat{\alpha}_1, \ldots, \hat{\alpha}_{1000}$.
- We estimate $\mathrm{Var}(\hat{\alpha})$ by

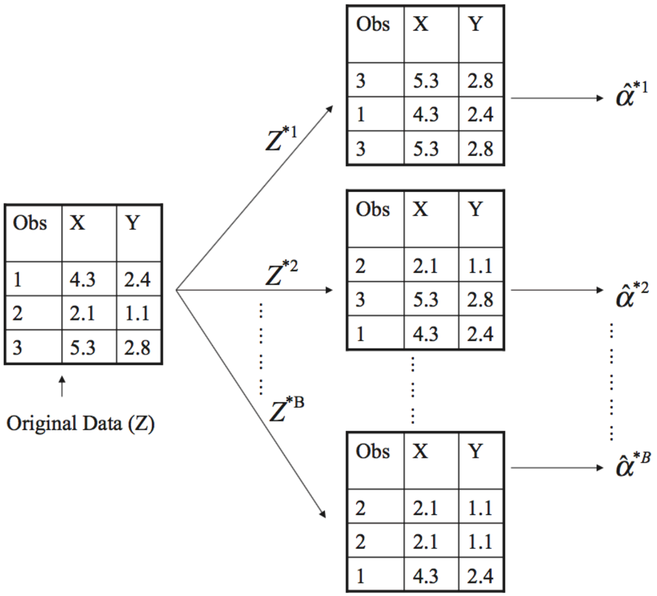
$$\frac{1}{1000-1}\sum_{r=1}^{1000}(\hat{\alpha}_r - \bar{\alpha})^2, \quad \text{where} \quad \bar{\alpha} = \frac{1}{1000}\sum_{r=1}^{1000}\hat{\alpha}_r.$$

Q: Is this feasible in practice?

Bootstrap

- The bootstrap approach re-samples from the original data set to mimic the process of obtaining new data sets in order to quantifty the uncertainty of a given procedure.
- Specifically, for a specified B (for instance, $B = 1000$) number of repetitions, we repeatedly sample the same amount of observations from the original data set with replacement.
- As a result, data set from bootstrap might contain some observations more than once, or zero time.

Simple illustration of Bootstrap



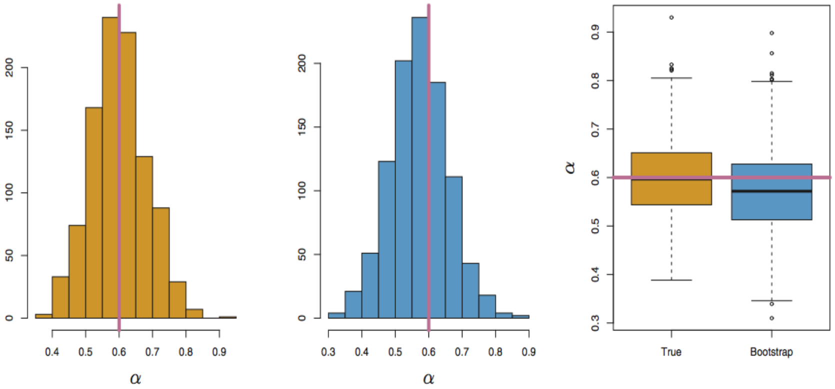
Bootstrap

Now let us apply bootstrap to estimate $\mathrm{Var}(\hat{\alpha})$:

- We denote the first bootstrap data set by Z^{*1}, and use Z^{*1} to construct the estimate of α, denoted by $\hat{\alpha}^{*1}$.
- This procedure is repeated B (say, $B = 1000$): specifically we simulate B different bootstrap data sets, $Z^{*1}, \dots, Z^{*B}$ and B corresponding estimates $\hat{\alpha}^{*1}, \dots, \hat{\alpha}^{*B}$.
- We estimate $\mathrm{Var}(\hat{\alpha})$ by the sample variance of $\hat{\alpha}^{*1}, \dots, \hat{\alpha}^{*B}$:

$$\frac{1}{B-1} \sum_{b=1}^{B} \left(\hat{\alpha}^{*b} - \bar{\alpha}^{*}\right)^2, \quad \text{where} \quad \bar{\alpha}^{*} = \frac{1}{B} \sum_{b=1}^{B} \hat{\alpha}^{*b}.$$

Example



- Left: The histogram of estimates of α obtained by generating 1,000 simulated data sets from the true population.
- Center: The histogram of estimates of α obtained from 1,000 bootstrap samples from a single data set.
- Right: Boxplots for estimates of α displayed in the left and center panels.

Bootstrap for quantifying the uncertainty of the OLS estimator

Given the data $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, the OLS gives

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

Statistical property of $\hat{\boldsymbol{\beta}} \in \mathbb{R}^p$ consists of

- its mean
- its covariance
  - *Recall that analyses of $\hat{\boldsymbol{\beta}}$ such as its mean and covariance are only available under linear model assumption.*
- its higher moments
- its whole distribution

Bootstrap can be used to estimate all above!

Bootstrap for quantifying the uncertainty of the OLS estimator

Given the data $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, the OLS gives

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

For $b = 1, \dots, B$,

1. obtain the bootstrap sample $D^b = (\mathbf{X}^b, \mathbf{y}^b)$
2. compute $\hat{\boldsymbol{\beta}}^b = (\mathbf{X}^{b\top} \mathbf{X}^b)^{-1} \mathbf{X}^{b\top} \mathbf{y}^b$.

Now for $\hat{\beta}_j$, the bootstrap estimates $\hat{\beta}_j^1, \dots, \hat{\beta}_j^B$ serve as "samples" of $\hat{\beta}_j$.

Bootstrap for quantifying the uncertainty of the OLS estimator

For instance,

- the mean of $\hat{\beta}_j$ can be estimated by

$$\frac{1}{B}\sum_{b=1}^{B}\hat{\boldsymbol{\beta}}_j^b.$$

- the variance of $\hat{\beta}_j$ can be estimated by

$$\frac{1}{B-1}\sum_{b=1}^{B}\left(\hat{\boldsymbol{\beta}}_j^b - \frac{1}{B}\sum_{b=1}^{B}\hat{\boldsymbol{\beta}}_j^b\right)^2.$$

- You can also estimate quantiles of the distribution of $\hat{\beta}_j$.
- In fact, you can estimate the whole distribution of $\hat{\beta}_j$.

The tutorial on Dec 4th will demonstrate the usage of bootstrap in R.

Bagging

- **Bootstrap aggregation**, or **bagging**, is a general-purpose procedure for reducing the variance of a statistical learning method;
- We introduce it here because it is particularly useful and frequently used in the context of decision trees.

Why bagging? The idea of bagging.

- Given a set of N independent estimates $\hat{\beta}^{(1)}, \ldots, \hat{\beta}^{(N)}$ of β, suppose

  $$\left|\mathbb{E}[\hat{\beta}^{(i)}] - \beta\right| \leq b, \qquad \mathrm{Var}(\hat{\beta}^{(i)}) = \sigma^2, \quad \forall\, i = 1, \ldots, N.$$

  Then the averaged estimate

  $$\bar{\beta} := \frac{1}{N} \sum_{i=1}^{N} \hat{\beta}^{(i)}$$

  satsifying

  $$\left|\mathbb{E}[\bar{\beta}] - \beta\right| \leq b, \qquad \mathrm{Var}(\bar{\beta}) = \frac{\sigma^2}{N}.$$

- In other words, averaging a set of independent random variables reduces the variance, meanwhile does not incur extra bias.

Bagging

- To apply bagging to regression trees, if we had B independent training sets, then we simply construct B regression trees using each of the training set, and average the resulting predictions in the end.
- However, we only have one training set. Instead, we sample with replacement the same amount of data points from the training data set B times. These sampled B data sets are called **bootstrap** samples.
- We train a decision tree by using the bth boostrap data and get the prediction $\hat{f}^{*b}(x)$ at a point x. We then average all the predictions by bagging

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

Bagging Trees

- The fitted decision trees on boostrap data sets are grown deep, and need not be pruned. Hence each individual tree has high variance, but low bias. Averaging these trees reduces the variance.

- Bagging can be also applied to classification trees. For a given test observation, we can record the class predicted by each of the B trees, and take a majority vote.

Out-of-Bag Error Estimation

- There is a simple way to estimate the test error of a bagged model, without the need to perform cross-validation or the validation set approach.
- For each bootstrap data set,
  - It only contains, on average, around $2/3$ of the original observations.
  - The remaining $1/3$ is not used to fit the decision tree, and is referred to as the **out-of-bag (OOB) observations**.
  - We predict the OOB observations by using the fitted decision tree.

Out-of-Bag Error Estimation

- For each observation i, we can pool its predictions across all B bootstrap data sets where i belongs to the OOB observations.
  - For regression, we average
  - For classification, we vote by majority
- We compute the MSE / misclassification error between predictions and the true responses over all observations.

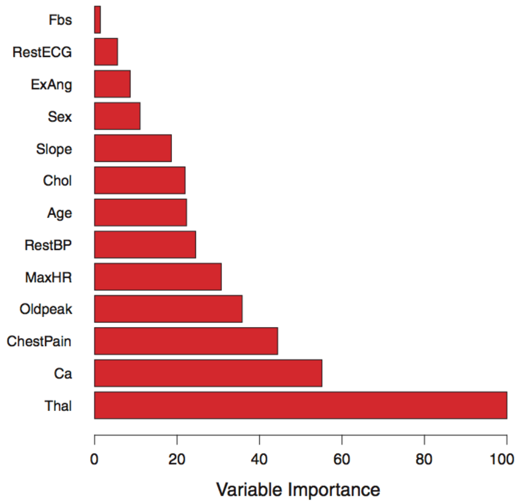
Variable Importance Measures

- Bagging typically reduces variance (i.e. improve accuracy) over prediction using a single decision tree.
- However, it can be difficult to interpret the resulting model.
- One can obtain an overall summary of the importance of each predictor using the RSS (for bagging regression trees) or the Gini index (for bagging classification trees).

Variable Importance Measures

- In the case of bagging regression trees, we can record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees.
  In bagging classification trees, we replace RSS by the Gini index.
- A large value of decreased RSS (or Gini index) indicates an important predictor.
- A graphical representation of variable importance is easy to draw.

Heart Data



Variable importance is computed using the mean decrease in Gini index, and expressed relative to the maximum.

Summary on bagging

- Bagging simply averages multiple decision trees
- Bias does not increase
- Variance is reduced
    - **How much is variance reduced?**

Random Forests

- Bagging is affected by the correlation among decision trees!
- **Random forests** provide an improvement over bagged trees by decorrelateing the trees.
  This reduces the variance when we average the trees.

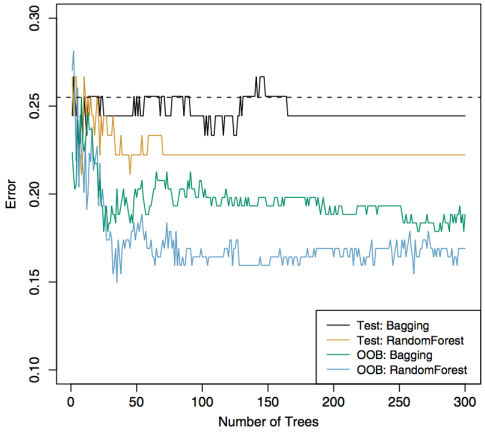
Random Forests

- As in bagging, we build a number of decision trees on bootstrap training samples.
- But for building these decision trees, when each time a split in a tree is considered, a random selection of $m < p$ predictors is chosen as split candidates from the full set of p predictors.
- A fresh selection of m predictors is taken at each split, and a common choice of m is $m \approx \sqrt{p}$.
- Random forest with $m = p$ is just bagging!

Why does Random Forest Reduce Correlations?

- Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors.
- Among all single decision trees, most of them will use this strong predictor in the top split. All of the single trees will look quite similar and could have high correlation.
- Random forests reduce the correlation by randomly selecting a subset of the predictors in each split of buiding each single tree. Indeed, on average many splits will not even consider the strong predictor.

Heart Data

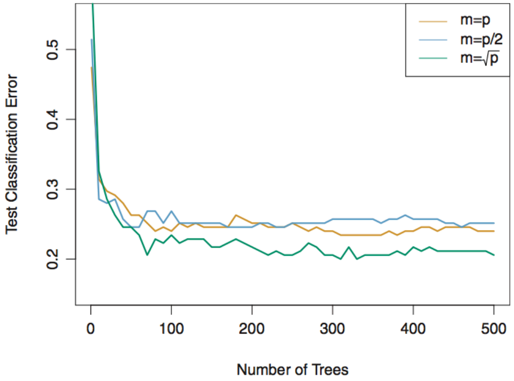


The test error (black and orange) is shown as a function of B, the number of bootstrapped training sets. Random forests were applied with $m = \sqrt{p}$. The dashed line indicates the test error resulting from a single classification tree. The green and blue traces show the OOB error, which is considerably lower.

Gene Expression Data

- We applied random forests to a high-dimensional biological data set consisting of expression measurements of 4,718 genes measured on tissue samples from 349 patients.
- There are around 20,000 genes in humans, and individual genes have different levels of activity, or expression, in particular cells, tissues, and biological conditions.
- Each of the patient samples has a qualitative label with 15 different levels: either normal or one of 14 different types of cancer.
- We use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set.
- We randomly divided the observations into a training and a test set, and applied random forests to the training set for three different values of the number of splitting variables m.

Gene Expression Data



The test errors are displayed as functions of the number of trees (B).

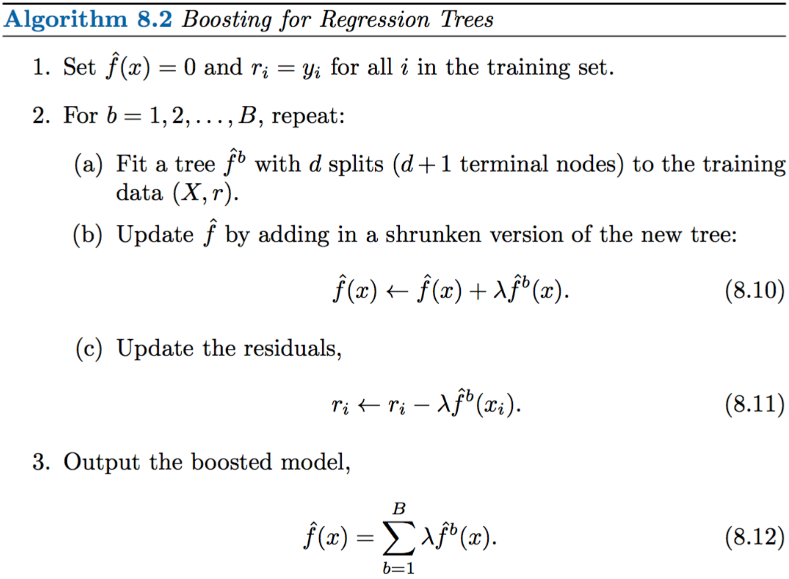
Random forests ($m < p$) lead to a slight improvement over bagging ($m = p$).

A single classification tree has an error rate of 45.7%.

Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression and classification. We focus on the context of decision trees.
- Recall that bagging (and random forest) involves creating multiple copies of the original training data set using bootstrap, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model.
- Boosting works both similarly and also differently
  - a Boosting combines different fitted trees
  - b the trees in boosting are grown sequentially: each tree is grown using information from previously grown trees
  - b Boosting does not involve bootstrap sampling; instead each tree is fitted on a modified version of the original data set.

Boosting



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

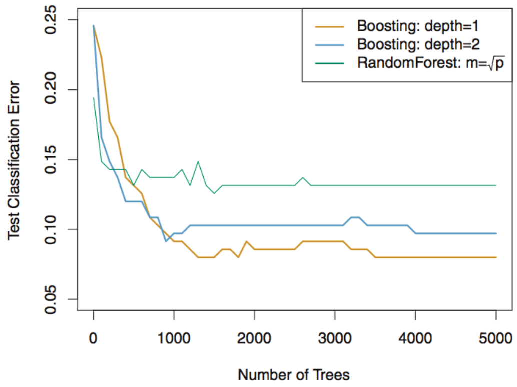
Boosting

- Unlike fitting a single large decision tree to the data, which might lead to overfitting, the boosting approach instead learns slowly.
- Given the current model, we fit a decision tree to the residuals from the model. We then add this new decision tree into the fitted function and update the residuals.
- Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm.
- By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well. The step size λ further slows down the learning process.

Tuning parameters for Boosting

- 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 if at all. We use cross-validation to select B.
- The shrinkage parameter (step size) λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001.
- The number of splits d in each tree, which controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree is a **stump**, consisting of a single split and resulting in an additive model.

Gene Expression Data



For both boosting trees, $\lambda = 0.01$. Depth-1 trees slightly outperform depth-2 trees and random forests. But these differeces are within their standard errors.

Summary

- Decision trees are simple and interpretable models for regression and classification.
- However their prediction accuracy is often not competitive with other regression / classification methods.
- Bagging, random forests and boosting can improve prediction accuracy of trees. They work by growing many trees on the training data and then combining the predictions of the resulting trees.
- The latter two methods – random forests and boosting – are among the state-of-the-art methods for supervised learning. However their results can be difficult to interpret.