DATA 300

Topic 6: Tree-based methods

Agenda

- Decision trees
  - Overview
  - Regression trees
  - Tree pruning
  - Classification trees
- Tree-based methods
  - Bagging
  - Random Forest
  - Boosting

Pros and cons of decision trees

- ▲ Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- ▲ Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.
- ▲ Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- ▲ Trees can easily handle qualitative predictors without the need to create dummy variables.

- ▼ Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.
- ▼ Additionally, trees can be very non-robust. In other words, a small change in the data can cause a large change in the final estimated tree.

All tree-based methods are trying to tackle these disadvantages.

General rule in reducing variance

If a model built on training set X might have high variance, we could build multiple models on multiple training sets $X_1, \ldots, X_K$, and average the performance over all models.

However, multiple training sets are often unavailable in practice.

Recall bootstrapping as a sampling methods

The Bootstrapping method refers to the procedure of repeatedly sampling (with replacement) from the dataset.

As a result, we could obtain *multiple datasets* from the initial dataset, and would be able to measure variation among datasets.

Bagging

Bagging (also referred to as *bootstrap aggregation*) is an application of bootstrapping methods to improve the performance of decision trees.

In bagging, we:

1) Use bootstrapping to create *B* training sets.
2) Using each training set $b = 1, \dots, B$, we build a decision tree f^b.
3) We then average all the predictions given by the B decision trees and obtain:

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

- This *average* is replaced by *majority vote* for classification problems.

Out-of-bag error

Unlike most methods where we use cross-validation to estimate the test error of a model, the test error could be estimated by the *out-of-bag* error in bagging.

Out-of-bag observation: mathematically, every observation is, on average, only included in two-thirds of the B bootstrapped training sets. In other words, it is *out-of-bag* for the remaining one-third.

Out-of-bag error

Out-of-bag observation: mathematically, every observation is, on average, only included in two-thirds of the B bootstrapped training sets. In other words, it is *out-of-bag* for the remaining one-third.

Therefore, to obtain the average *out-of-bag error:*

- For each tree f^b built with sample set b, we use this tree to estimate the sample units that were not part of set b.
- We then average across all trees.

This is equivalent to leave-one-out cross-validation error.

Variable Importance Measures

Because bagging is an average over multiple decision trees, it is nearly impossible to interpret the results. Bagging reduces the variance of decision trees at the expensive of interpretability.

Variable Importance Measures

Because bagging is an average over multiple decision trees, it is nearly impossible to interpret the results. Bagging reduces the variance of decision trees at the expensive of interpretability.

Therefore, to measure the importance of different variables, we have to track the *average* impact of each variable over B trees. Specifically, for each predictor, we track the *total decreased RSS/GINI index/entropy.*

Variable importance are then sorted by their contribution to RSS/Gini/entropy.

Summary of Bagging

In bagging, we:

1) Use bootstrapping to create *B* training sets.
2) Using each training set $b = 1, \dots, B$, we build a decision tree f^b.
3) We then average all the predictions given by the B decision trees and obtain:

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

   - This *average* is replaced by *majority vote* for classification problems.

The *out-of-bag error* is used in replace of a cross-validation error to approximate test error.

Variable importance is sorted by their contribution in decreasing RSS/GINI/Entropy.

Summary of Bagging

In bagging, we:

1) Use bootstrapping to create *B* training sets.
2) Using each training set $b = 1, \dots, B$, we build a decision tree f^b.
3) We then average all the predictions given by the B decision trees and obtain:

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

- This *average* is replaced by *majority vote* for classification problems.

What happens when there is a strong predictor?

- Would all trees include use this predictor to split?
- If so, how does it impact the advantage of variance reduction brought by bagging?

Summary of Bagging

In bagging, we:

1) Use bootstrapping to create *B* training sets.
2) Using each training set $b = 1, \dots, B$, we build a decision tree f^b.
3) We then average all the predictions given by the B decision trees and obtain:

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

   - This *average* is replaced by *majority vote* for classification problems.

What happens when there is a strong predictor?

- Would all trees include use this predictor to split?
  - Yes, as a result, all trees will look similar.
- If so, how does it impact the advantage of variance reduction brought by bagging?
  - Bagging no longer serves its purpose of variance reduction, because all trees are correlated.

Agenda

- Decision trees
  - Overview
  - Regression trees
  - Tree pruning
  - Classification trees
- Tree-based methods
  - Bagging
  - **Random Forest**
  - Boosting

Random Forest

In bagging, we:

1) Use bootstrapping to create *B* training sets.
2) Using each training set $b = 1, \dots, B$, we build a decision tree f^b.
3) We then average all the predictions given by the B decision trees and obtain: $\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$
   - This *average* is replaced by *majority vote* for classification problems.

In the scenario where there is a strong predictor, how do you avoid trees *all* picking this predictor?

Random Forest

In bagging, we:

1) Use bootstrapping to create *B* training sets.
2) Using each training set $b = 1, \dots, B$, we build a decision tree f^b.
3) We then average all the predictions given by the B decision trees and obtain: $\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$
   - This *average* is replaced by *majority vote* for classification problems.

In the scenario where there is a strong predictor, how do you avoid trees *all* picking this predictor?

- When building a tree, only give this tree a *random* sample of m predictors (assuming there is a total of *p* predictors, and $m = \sqrt{p}$)

Random Forest

Similar to bagging, random forest also start by using bootstrapping to create *B* training sets. However:

1) For each training set $b = 1, \dots, B$, random forest only takes a random sample of m $(m = \sqrt{p})$ predictors to build a decision tree f^b.
2) We then average all the predictions given by the B decision trees and obtain:

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

- This *average* is replaced by *majority vote* for classification problems.

Agenda

- Decision trees
  - Overview
  - Regression trees
  - Tree pruning
  - Classification trees
- Tree-based methods
  - Bagging
  - Random Forest
  - **Boosting**

Boosting

Both bagging and random forest are building multiple trees, where each tree is a model with predictors X and response y.

Boosting, on the other hand, builds **small** trees where each tree is a model between predictors X and the residual r.

Boosting

Both bagging and random forest are building multiple trees, where each tree is a model with predictors X and response y.

Boosting, on the other hand, builds **small** trees where each tree is a model between predictors X and the residual r.

- At step 1, the residual r = y. We build a tree with *d* splits (d = 1 usually works well enough).

Boosting

Both bagging and random forest are building multiple trees, where each tree is a model with predictors X and response y.

Boosting, on the other hand, builds **small** trees where each tree is a model between predictors X and the residual r.

- At step 1, the residual r = y. We build a tree f^1 with *d* splits (d = 1 usually works well enough).
  - Part of the response y is explained by this tree $\widehat{f^1}$, so residuals become

$$r \leftarrow r - \lambda \widehat{f^1}$$

    where λ controls the *learning* speed. λ often equals to 0.01 or 0.001.
  - We only need to focus on the residuals going forward.

Boosting

Both bagging and random forest are building multiple trees, where each tree is a model with predictors X and response y.

Boosting, on the other hand, builds **small** trees where each tree is a model between predictors X and the residual r.

- At step 1, the residual r = y. We build a tree f^1 with *d* splits (d = 1 usually works well enough).
  - Part of the response y is now explained by this tree $\widehat{f^1}$, so residuals become

$$r \leftarrow r - \lambda\widehat{f^1}$$

  where λ controls the *learning* speed. λ often equals to 0.01 or 0.001.
- At step 2, residual is now r = y $- \lambda\widehat{f^1}$. We again build a tree f^2, update the residuals.

Boosting

Both bagging and random forest are building multiple trees, where each tree is a model with predictors X and response y.

Boosting, on the other hand, builds **small** trees where each tree is a model between predictors X and the residual r.

- At step 1, the residual r = y. We build a tree f^1 with *d* splits (d = 1 usually works well enough).
  - Part of the response y is now explained by this tree $\widehat{f^1}$, so residuals become

    $$r \leftarrow r - \lambda\widehat{f^1}$$

    where λ controls the *learning* speed. λ often equals to 0.01 or 0.001.
- At step 2, residual is now r = y $- \lambda\widehat{f^1}$. We again build a tree f^2, update the residuals.
- Assuming we continue until we built B trees, the final Boosting model becomes:

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

Agenda

- Decision trees
  - Overview
  - Regression trees
  - Tree pruning
  - Classification trees
- **Tree-based methods**
  - Bagging
  - Random Forest
  - Boosting

Summary of tree-based methods

- Bagging: Obtain B bootstrapped datasets and build B trees independently. The final model is an average (majority vote) over all B trees.

Summary of tree-based methods

- Bagging: Obtain B bootstrapped datasets and build B trees independently. The final model is an average (majority vote) over all B trees.
- Random Forest: Obtain B bootstrapped datasets and build B trees independently. Each tree only uses a random set of $\sqrt{p}$ predictors, assuming p is the total number of predictors available.

Summary of tree-based methods

- Bagging: Obtain B bootstrapped datasets and build B trees independently. The final model is an average (majority vote) over all B trees.
- Random Forest: Obtain B bootstrapped datasets and build B trees independently. Each tree only uses a random set of $\sqrt{p}$ predictors, assuming p is the total number of predictors available.
- Boosting: Each tree is built sequentially such that the algorithm fits to the residuals from previous trees.