Lecture 08
Decision Trees II

15 February 2016

Taylor B. Arnold
Yale Statistics
STAT 365/665

Yale

Updated office hours:

- Taylor Arnold Wednesdays, 16:00 - 17:00, HH 24
- Elena Khusainova Tuesdays, 13:00-15:00, HH 24, Basement
- Yu Lu Wednesdays, 19:00-20:30, HH 24, Basement
- Jason Klusowski Thursdays, 19:00-20:30, HH 24

Problem Set 2:

- We will give you full credit either way you approach this, but your solution for Question #4 will be much more interesting if you weight the MSE of the census tracts by their population

Problem Set 1:

- Finished grading them; will likely post grades to ClassesV2 by tomorrow morning

Problem Set 1:

- Finished grading them; will likely post grades to ClassesV2 by tomorrow morning
- Actual grades were okay; median of 9/10, mean of 8.5/10

Problem Set 1:

- Finished grading them; will likely post grades to ClassesV2 by tomorrow morning
- Actual grades were okay; median of 9/10, mean of 8.5/10
- However, many people not following directions:
  - not putting files in a '.zip' directory
  - using a different zip format (.rar, .7z, .tar.gz)
  - naming files incorrectly
  - including column names, quotes, or both in the set of predicted values

Problem Set 1:

- Finished grading them; will likely post grades to ClassesV2 by tomorrow morning
- Actual grades were okay; median of 9/10, mean of 8.5/10
- However, many people not following directions:
  - not putting files in a '.zip' directory
  - using a different zip format (.rar, .7z, .tar.gz)
  - naming files incorrectly
  - including column names, quotes, or both in the set of predicted values
- As well, many of the R and Python scripts did not actually run correctly; a common problem being that they threw errors when they could not find global variables or local files

Problem Set 1, cont.

- Paying attention to these details is very important!
- In almost all cases, you got full credit for the implementation and data analysis questions as long as you wrote *something* and your code ran without any errors
- Future problem sets will likely have fewer 'parts' to them, but the actually content will be graded more strictly, so you do not want to lose points on silly things
- Test your code!
- Remember:
  - No late submissions (unless you have a true emergency, and for undergraduates a Deans excuse)
  - We do drop the lowest grade from the final assignment

- Today:
  - Finish decision trees
  - Walk through an analysis of trees for classification and regression
  - Introduction to SVMs (time permitting)

Gradient boosted trees

Last time, we looked at decision trees and random forests. There is one additional variant on these ideas which has been shown to produce extremely powerful predictions. Called gradient boosted trees.

Gradient boosted trees, cont.

The algorithm for this can conceptually be described as follows:

1. Fit a decision tree $T^{(1)}$ on a randomly sampled subset of the data (but using all of the variables at each node, unlike RF) and call the predicted values from this tree on the entire dataset $\widehat{y}^{(1)}$.

Gradient boosted trees, cont.

The algorithm for this can conceptually be described as follows:

1. Fit a decision tree $T^{(1)}$ on a randomly sampled subset of the data (but using all of the variables at each node, unlike RF) and call the predicted values from this tree on the entire dataset $\widehat{y}^{(1)}$.
2. Calculate the residuals $r^{(1)} = y - \widehat{y}^{(1)}$.

Gradient boosted trees, cont.

The algorithm for this can conceptually be described as follows:

1. Fit a decision tree $T^{(1)}$ on a randomly sampled subset of the data (but using all of the variables at each node, unlike RF) and call the predicted values from this tree on the entire dataset $\widehat{y}^{(1)}$.
2. Calculate the residuals $r^{(1)} = y - \widehat{y}^{(1)}$.
3. Fit a new decision tree $T^{(2)}$ on a random resampling of the residuals $r^{(1)}$, and calculate the predicted values $\widehat{r}^{(1)}$ for the entire dataset.

Gradient boosted trees, cont.

The algorithm for this can conceptually be described as follows:

1. Fit a decision tree $T^{(1)}$ on a randomly sampled subset of the data (but using all of the variables at each node, unlike RF) and call the predicted values from this tree on the entire dataset $\widehat{y}^{(1)}$.
2. Calculate the residuals $r^{(1)} = y - \widehat{y}^{(1)}$.
3. Fit a new decision tree $T^{(2)}$ on a random resampling of the residuals $r^{(1)}$, and calculate the predicted values $\widehat{r}^{(1)}$ for the entire dataset.
4. Now, set $\widehat{y}^{(2)}$ to be $\widehat{y}^{(1)} + \widehat{r}^{(1)}$ and set the new residuals to be $r^{(2)} = y - \widehat{y}^{(2)}$.

Gradient boosted trees, cont.

The algorithm for this can conceptually be described as follows:

1. Fit a decision tree $T^{(1)}$ on a randomly sampled subset of the data (but using all of the variables at each node, unlike RF) and call the predicted values from this tree on the entire dataset $\widehat{y}^{(1)}$.
2. Calculate the residuals $r^{(1)} = y - \widehat{y}^{(1)}$.
3. Fit a new decision tree $T^{(2)}$ on a random resampling of the residuals $r^{(1)}$, and calculate the predicted values $\widehat{r}^{(1)}$ for the entire dataset.
4. Now, set $\widehat{y}^{(2)}$ to be $\widehat{y}^{(1)} + \widehat{r}^{(1)}$ and set the new residuals to be $r^{(2)} = y - \widehat{y}^{(2)}$.
5. Calculate decision tree $T^{(3)}$ on a randomly sampled set of the residuals $r^{(2)}$, and calculate the predicted values $\widehat{r}^{(2)}$ for the entire dataset.

Gradient boosted trees, cont.

The algorithm for this can conceptually be described as follows:

1. Fit a decision tree $T^{(1)}$ on a randomly sampled subset of the data (but using all of the variables at each node, unlike RF) and call the predicted values from this tree on the entire dataset $\widehat{y}^{(1)}$.
2. Calculate the residuals $r^{(1)} = y - \widehat{y}^{(1)}$.
3. Fit a new decision tree $T^{(2)}$ on a random resampling of the residuals $r^{(1)}$, and calculate the predicted values $\widehat{r}^{(1)}$ for the entire dataset.
4. Now, set $\widehat{y}^{(2)}$ to be $\widehat{y}^{(1)} + \widehat{r}^{(1)}$ and set the new residuals to be $r^{(2)} = y - \widehat{y}^{(2)}$.
5. Calculate decision tree $T^{(3)}$ on a randomly sampled set of the residuals $r^{(2)}$, and calculate the predicted values $\widehat{r}^{(2)}$ for the entire dataset.
6. Repeat as needed until you have predictions $\widehat{y}^{(K)}$.

Gradient boosted trees, cont.

The actual algorithm is modified it two ways:

1. The predicted values in the terminal nodes of the decision tree $T^{(j)}$ for any j are not determined just by the sampled set of data, but rather chosen to minimize the actual loss of using the $\widehat{y}^{(j)}$.
2. The values for the predicted residuals $\widehat{r}^{(j)}$ is 'shrunk' towards zero by a factor of $0 < \rho \leq 1$. This value is called the learning rate.

Due to the learning rate factor, it makes sense to center the input variables before applying this algorithm (this is usually done by the software).