Random Forests

Ryan Miller

Introduction

- Classification and regression trees (CART) provide a sensible, easily interpreted model for scenarios involving highly interactive or non-linear sets of explanatory variables
  - Unfortunately, CART models tend to have high variance, making them prone to overfitting

Introduction

- Classification and regression trees (CART) provide a sensible, easily interpreted model for scenarios involving highly interactive or non-linear sets of explanatory variables
  - Unfortunately, CART models tend to have high variance, making them prone to overfitting
- In the well-switching example, our CART model had an *in-sample accuracy* of 63.1% but its *cross-validated accuracy* was only 61.2%
  - For comparison, logistic regression (with interaction) had *in-sample accuracy* of 62.4% and a *cross-validated accuracy* of 62.2%

Random Forests and Variance

- Random forests are application of tree-based models centering upon the idea that the average of a set has lower variance than the individual observations

Random Forests and Variance

- Random forests are application of tree-based models centering upon the idea that the average of a set has lower variance than the individual observations
- This concept is seen extensively in classical statistics. Suppose a random variable, X, is normally distributed as follows: $X \sim N(\mu, \sigma)$
  - By CLT, we know $Var(\bar{x}) = \sigma^2/n$, while $Var(x_1) = \sigma^2$

Random Forests and Variance

- A similar idea applies to predictive models
    - If several separate predictive models are averaged, the result will have lower variance (less propensity towards overfitting) than any of the individual models

Random Forests and Variance

- A similar idea applies to predictive models
  - If several separate predictive models are averaged, the result will have lower variance (less propensity towards overfitting) than any of the individual models
- Random forests exploit this fact by averaging the predictions of many different CART models to obtain a single, low-variance prediction
  - The challenge in doing this is that the models need to be independent of each other. . .

Bootstrapping

- Bootstrapping is a general statistical approach used to mimic the generation of new data
  - The main idea is to *randomly sample* the original dataset *with replacement* to construct a *bootstrapped sample*
  - Often, the process is repeated many times to create a set of B unique bootstrap samples

Bootstrapping

- Bootstrapping is a general statistical approach used to mimic the generation of new data
  - The main idea is to *randomly sample* the original dataset *with replacement* to construct a *bootstrapped sample*
  - Often, the process is repeated many times to create a set of B unique bootstrap samples

```r
set.seed(123)
n <- length(Wells)
B <- numeric(10)

## Bootstrapping the mean arsenic level (10 different bootstrap samples)
for(i in 1:length(B)){
  boot_idx <- sample(1:n, size = n, replace = TRUE)
  boot_sample <- Wells[boot_idx,]
  B[i] <- mean(boot_sample$arsenic)
}
B
```

```
##  [1] 1.526 1.478 1.750 1.720 1.730 1.226 1.730 1.468 1.032 2.050
```

The Random Forest Algorithm

1) Create B bootstrap samples
2) For bootstrap sample, fit a CART model, but do so by randomly selecting a subset of m predictors to be considered at each split
3) Each of the B trees in the forest contributes a prediction or "vote", with the majority (or average) of these votes forming the random forest's final prediction

The Random Forest Algorithm

1) Create B bootstrap samples
2) For bootstrap sample, fit a CART model, but do so by randomly selecting a subset of m predictors to be considered at each split
3) Each of the B trees in the forest contributes a prediction or "vote", with the majority (or average) of these votes forming the random forest's final prediction

Note:

- The random selection of m predictors to consider at each split prevents the same variables from always dominating every tree, which further decorrelates the predictions (or votes) of the different trees

The Random Forest Algorithm

Random forest models can be trained using the `randomForest` package in `R`:

```r
## Random forest (B = 10)
library(randomForest)
rf <- randomForest(switch ~ ., data = Wells, ntree = 10)

## Vote distribution (first for the first 6 data-points)
rf$votes[1:6,]
```

```
##          no       yes
## 1 0.0000000 1.0000000
## 2 0.7500000 0.2500000
## 3 0.0000000 1.0000000
## 4 0.0000000 1.0000000
## 5 0.6666667 0.3333333
## 6 0.0000000 1.0000000
```

Evaluating Random Forest Predictions

- Because bootstrapping will naturally omit some data-points from each bootstrap sample, cross-validation is not necessary to measure out-of-sample performance
  - Instead the “bagged” data-points can be used as test data, yielding “out of bag”, or OOB, performance measures

Evaluating Random Forest Predictions

- Because bootstrapping will naturally omit some data-points from each bootstrap sample, cross-validation is not necessary to measure out-of-sample performance
  - Instead the “bagged” data-points can be used as test data, yielding “out of bag”, or OOB, performance measures

```
## OOB error rates for each of our 10 trees
rf$err.rate
```

```
##              OOB        no       yes
##  [1,] 0.4354839 0.5170940 0.3765432
##  [2,] 0.4334433 0.5361757 0.3572797
##  [3,] 0.4365639 0.5051546 0.3853846
##  [4,] 0.4368088 0.5032377 0.3873191
##  [5,] 0.4351852 0.5051903 0.3827720
##  [6,] 0.4307036 0.5033223 0.3763975
##  [7,] 0.4369369 0.5056818 0.3857316
##  [8,] 0.4419002 0.5172969 0.3862151
##  [9,] 0.4431664 0.5226370 0.3842074
## [10,] 0.4273762 0.5102041 0.3658110
```

```
mean(rf$err.rate[,1])
```

```
## [1] 0.4357569
```

Evaluating Random Forest Predictions

With some minor tweaking of the *tuning parameters*, it's easy to find a model with out-of-sample accuracy superior to both logistic regression and CART

- `mtry` is the number of variables to randomly consider at each split
- `nodesize` is the minimum size of each terminal node

```
rf <- randomForest(switch ~ ., data = Wells, ntree = 500,
                   mtry = 2, nodesize   = 100)
1 - mean(rf$err.rate[,1])
```

```
## [1] 0.6360171
```

Random Forest Pros and Cons:

Pros:

- Better accuracy than most models
- Tends to work well even if the data includes outliers, non-linearity, interactions, and missing data

Cons:

- Significantly harder to interpret compared to individual trees
  - Methods have been developed to output the "average tree"
  - Methods have been developed to measure the "importance" of each variable

Closing Remarks

- Random forests are a very powerful predictive modeling approach, but that accuracy comes at the expense of interpretability
  - Unlike a single CART model (or even logistic regression), it's difficult to communicate how a random forest generates predictions
- However, random forests will generally yield better out-of-sample performance than both CART and logistic regression (given proper choices for tuning parameters such as `mtry` and `nodesize`)