Ensemble Methods

Nisha Iyer

April 1, 2016

### I will use the well known Boston Housing Data set to build out a random forest model, followed

### by an AdaBoost model. I want to use the same data set for comparison of these two methods.

data(Boston)

## Warning in data(Boston): data set 'Boston' not found

library(MASS)  
library(caret)

## Loading required package: lattice  
## Loading required package: ggplot2

library(randomForest)

## randomForest 4.6-12  
## Type rfNews() to see new features/changes/bug fixes.  
##   
## Attaching package: 'randomForest'  
##   
## The following object is masked from 'package:ggplot2':  
##   
## margin

library(ROCR)

## Loading required package: gplots  
##   
## Attaching package: 'gplots'  
##   
## The following object is masked from 'package:stats':  
##   
## lowess

library(miscTools)  
  
#load Boston Housing data   
#this will be regression trees since the target variable is continuous (median value)  
data(Boston)  
head(Boston)

## crim zn indus chas nox rm age dis rad tax ptratio black  
## 1 0.00632 18 2.31 0 0.538 6.575 65.2 4.0900 1 296 15.3 396.90  
## 2 0.02731 0 7.07 0 0.469 6.421 78.9 4.9671 2 242 17.8 396.90  
## 3 0.02729 0 7.07 0 0.469 7.185 61.1 4.9671 2 242 17.8 392.83  
## 4 0.03237 0 2.18 0 0.458 6.998 45.8 6.0622 3 222 18.7 394.63  
## 5 0.06905 0 2.18 0 0.458 7.147 54.2 6.0622 3 222 18.7 396.90  
## 6 0.02985 0 2.18 0 0.458 6.430 58.7 6.0622 3 222 18.7 394.12  
## lstat medv  
## 1 4.98 24.0  
## 2 9.14 21.6  
## 3 4.03 34.7  
## 4 2.94 33.4  
## 5 5.33 36.2  
## 6 5.21 28.7

dim(Boston)

## [1] 506 14

#use createDataPartition from the caret package to create train and test sets  
set.seed(12345)  
trainIndex <- createDataPartition(Boston$medv, p = .7, list=FALSE, times=1)  
head(trainIndex)

## Resample1  
## [1,] 1  
## [2,] 2  
## [3,] 4  
## [4,] 5  
## [5,] 6  
## [6,] 8

BostonTrain <- Boston[trainIndex,]  
BostonTest <- Boston[-trainIndex,]  
#A note about 'createDataPartition; createResample can be used to make a simple bootstrap and createFolds for   
#cross-validation groupings.  
  
#OK, training and test sets have been created. Now we will run randomForest on the training data.  
  
#The randomForest algorithm below includes target variable (medv) '.' which is all the predictors,  
#the data is BostonTrain, number of trees to create using 'bagging' method is 100 and what makes the   
#random forest algorithm set apart from boosting; number of predictors randomly chosen from full set of  
#predictors is 5. In regression trees, the recommended number for mtry is the total number of predictors  
#divided by three. In classification (an example to follow) the recommended number is the square root of  
#predictors.   
trainRandom <- randomForest(medv ~ ., BostonTrain,ntree = 100,mtry = 5)  
trainRandom

##   
## Call:  
## randomForest(formula = medv ~ ., data = BostonTrain, ntree = 100, mtry = 5)   
## Type of random forest: regression  
## Number of trees: 100  
## No. of variables tried at each split: 5  
##   
## Mean of squared residuals: 12.58898  
## % Var explained: 85.18

#since this is a regression tree, we will calculate r^2 and RMSE  
r2 <- rSquared(BostonTest$medv, BostonTest$medv - predict(trainRandom, BostonTest))  
r2

## [,1]  
## [1,] 0.8871649

rmse <- sqrt(mean(BostonTest$medv - predict(trainRandom, BostonTest))^2)  
rmse

## [1] 0.3486727

#The model seems to perform well. We can look at variable importance, which is based on out of bag error and   
#gain or loss of mean square error based on the predictor's power  
BostonImp <- importance(trainRandom, type = 1)