Decision Trees, Random Forest and XGBoost

Muhammad Tamjid Rahman

Contents

Loading R packages

```r
library(uuml)
library(xgboost)
library(randomForest)
```

1 Decision Trees

In this section we'll work with the data set "Hitters". The data set has 322 observations with 20 variables. There are 59 missing values in "Salary" column. We'll remove all rows containing "NA". Then the final data set has 263 observations.

1 Create a test set

We created a test set by setting aside the 30 first observations from the data set "Hitters". And we created a train data set with the rest.

```r
data("Hitters")
# Remove NA values
Hitters <- Hitters[complete.cases(Hitters),]
# Create test and training set
X_test <- Hitters[1:30, c("Years", "Hits")]
y_test <- Hitters[1:30, c("Salary")]
X_train <- Hitters[31:nrow(Hitters), c("Years", "Hits")]
y_train <- Hitters[31:nrow(Hitters), c("Salary")]
```

2

We implemented an R function to split observations binary greedily.

```r
# Lets choose a small data to try out our algorithm with
X_check <- Hitters[31:50, c("Years", "Hits")]
y_check <- Hitters[31:50, c("Salary")]
# These are the names of the players we look at
rownames(Hitters)[31:50]
```

```
##  [1] "-Bob Melvin"       "-BillyJo Robidoux" "-Bill Schroeder"
##  [4] "-Chris Bando"      "-Chris Brown"      "-Carmen Castillo"
##  [7] "-Chili Davis"      "-Carlton Fisk"     "-Curt Ford"
## [10] "-Carney Lansford"  "-Chet Lemon"       "-Candy Maldonado"
## [13] "-Carmelo Martinez" "-Craig Reynolds"   "-Cal Ripken"
## [16] "-Cory Snyder"      "-Chris Speier"     "-Curt Wilkerson"
## [19] "-Dave Anderson"    "-Don Baylor"
```

```
tree_split <- function(X, y, l){
    SS <- matrix(NA, nrow(X), ncol(X))
    S <- matrix(NA, nrow(X), ncol(X))
    for(j in 1:ncol(X)){
        for(k in 1:nrow(X)){
            s <- X[k,j]
            if(sum(X[,j] <= s) >= l & sum(X[,j] > s) >= l){
                c1 <- mean(y[X[,j] <= s])
                c2 <- mean(y[X[,j] > s])
                SS[k,j] <- sum((y[X[,j] <= s] - c1)^2) + sum((y[X[,j] > s] - c2)^2)
                S[k,j] <- s
            }
            else{
                SS[k,j] <- Inf
                S[k,j] <- NA
            }
        }
    }
    i <- which(SS == min(SS), arr.ind = TRUE)[1,1]
    j <- which(SS == min(SS), arr.ind = TRUE)[1,2]
    s_final <- S[i, j]
    R1 <- which(X[,j] <= s_final)
    R2 <- which(X[,j] > s_final)

    return(list(j = j,
                s = s_final,
                R1 = R1,
                R2 = R2,
                SS = min(SS)))
}
```

We made a split without any limit on leaf size

```
tree_split(X_check, y_check, l = 1)
```

```
## $j
## col
##   2
##
## $s
## [1] 132
##
## $R1
##  [1]  1  2  3  4  5  6  8  9 11 12 13 14 16 17 18 19
##
## $R2
## [1]  7 10 15 20
```

```
##
## $SS
## [1] 904383.4
```

Then We made a split without on leaf size=5

```
tree_split(X_check, y_check, l = 5)
```

```
## $j
## col
##   1
##
## $s
## [1] 5
##
## $R1
##  [1]  1  2  3  5  6  9 13 16 18 19
##
## $R2
##  [1]  4  7  8 10 11 12 14 15 17 20
##
## $SS
## [1] 1346633
```

3 The frst split based on the whole training data X_train and y_train

```
tree_split(X_train, y_train, l = 5)$R1
```

```
##  [1]   1   2   3   5   9  13  16  18  19  21  22  28  30  36  38  40  41  42  45
## [20]  51  55  57  73  75  78  79  80  88  89  94  96  99 100 102 104 107 113 114
## [39] 118 119 121 128 130 133 134 138 139 141 142 143 145 146 147 149 151 152 155
## [58] 157 167 178 180 183 184 185 187 191 192 193 194 197 198 199 204 206 210 211
## [77] 216 220 222 227 228
```

```
tree_split(X_train, y_train, l = 5)$R2
```

```
##   [1]   4   6   7   8  10  11  12  14  15  17  20  23  24  25  26  27  29  31
##  [19]  32  33  34  35  37  39  43  44  46  47  48  49  50  52  53  54  56  58
##  [37]  59  60  61  62  63  64  65  66  67  68  69  70  71  72  74  76  77  81
##  [55]  82  83  84  85  86  87  90  91  92  93  95  97  98 101 103 105 106 108
##  [73] 109 110 111 112 115 116 117 120 122 123 124 125 126 127 129 131 132 135
##  [91] 136 137 140 144 148 150 153 154 156 158 159 160 161 162 163 164 165 166
## [109] 168 169 170 171 172 173 174 175 176 177 179 181 182 186 188 189 190 195
## [127] 196 200 201 202 203 205 207 208 209 212 213 214 215 217 218 219 221 223
## [145] 224 225 226 229 230 231 232 233
```

4 The Sum of Squares (SS) for this frst split

```
tree_split(X_train, y_train, l = 5)$SS
```

```
## [1] 38464163
```

5 Creating a grow_tree() function

```r
grow_tree <- function(X, y, l){
    m = 1
    M = 1
    S <- list(1: nrow(X))
    j <- c()
    s <- c()
    gamma <- c()
    R1_i <- c()
    R2_i <- c()
    y1 <-list(0)
    x1 <-list(0)
    y1[[1]]<-y
    x1[[1]]<-X
    while(m <= M){
        if(length(S[[m]]) >= 2*l){
            split <- tree_split(x1[[m]],y1[[m]],l)
            S[[M+1]] <- split$R1
            S[[M+2]] <- split$R2

            j[m] <- split$j
            s[m] <- split$s
            gamma[m] <- NA
            R1_i[m] = M+1
            R2_i[m] = M+2
            y1[[M+1]]=y1[[m]][S[[M+1]]]
            x1[[M+1]]=x1[[m]][S[[M+1]],]
            y1[[M+2]]=y1[[m]][S[[M+2]]]
            x1[[M+2]]=x1[[m]][S[[M+2]],]
            M = M + 2
        }
        else{
            gamma[m] <- mean(y1[[m]],na.rm=T)

            j[m] <- NA
            s[m] <- NA
            R1_i[m] <- NA
            R2_i[m] <- NA
        }
        m = m + 1
    }
    return(data.frame(j = j,
                     s = s,
                      R1_i = R1_i,
                      R2_i = R2_i,
                      gamma = gamma))
}

tr <- grow_tree(X_check, y_check, l = 5)
tr
```

```
##    j   s R1_i R2_i    gamma
## 1  1   5    2    3       NA
## 2  1   3    4    5       NA
## 3  2 101    6    7       NA
## 4 NA   NA    NA    NA 106.5000
## 5 NA   NA    NA    NA 244.5000
## 6 NA   NA    NA    NA 509.3334
## 7 NA   NA    NA    NA 946.0000
```

6 Prediction

```
X_new <- Hitters[51:52, c("Years", "Hits")]
X_new
```

```
##                  Years Hits
## -Daryl Boston        3   53
## -Darnell Coles       4  142
```

```
y_new <- Hitters[51:52, c("Salary")]
y_new
```

```
## [1]  75 105
```

```
predict_with_tree <- function(new_data, tree){
    pred <- c()
    for(i in 1:nrow(new_data)){
        m <- 1
        x <- new_data[i, tree$j[m]]
        while(is.na(tree$j[m])==F){
            if(x <= tree$s[m]){
                m = tree$R1_i[m]
            }
            else{
                m = tree$R2_i[m]
            }
        }
        pred[i] <- tree$gamma[m]
    }
    return(pred)
}
```

```
pred_y <- predict_with_tree(new_data = X_new, tree = tr)
pred_y
```

```
## [1] 106.5 244.5
```

7 Implementing a rmse(x,y) function

```
rmse<- function(x,y){
    rmse=sqrt((1/length(x))*(sum((x-y)^2)))
    return(rmse)
}

rmse(c(75, 105), c(102, 99))
```

```
## [1] 19.55761
```

8 RMSE on the test set for a tree trained on the whole training data X_train and y_train

```r
tr1 <- grow_tree(X_train, y_train, l = 5)

y_pred <- predict_with_tree(new_data = X_test, tree = tr1)

rmse(y_test, y_pred)
```

```
## [1] 415.7744
```

2 Running standard tools for Boosting and Random Forest

1 Random Forest regression

```r
data("Hitters")
Hitters <- Hitters[complete.cases(Hitters),]
dat_test <- Hitters[1:30, c("Salary","Years", "Hits")]
dat_train <- Hitters[31:nrow(Hitters), c("Salary","Years", "Hits")]
Hitters.rf <- randomForest(Salary ~ Years + Hits, data = dat_train)
```

2

```r
Hitters.rf$mtry
```

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

The number of variables used at each split is 1. The default values are different for classification (sqrt(p) where p is number of variables in train data and regression (p/3). In our case, since p=2, one variable is used in each split.

3 Prediction

```r
y_pred <- predict(Hitters.rf, dat_test[,c('Years','Hits')])
y_pred <- as.numeric(y_pred)
# RMSE
rmse( dat_test[,c('Salary')], y_pred)
```

```
## [1] 237.7139
```

4 XGBoost

```r
X_test <- dat_test[1:30, c("Years", "Hits")]
y_test <- dat_test[1:30, c("Salary")]
X_train <-Hitters[31:nrow(Hitters), c("Years", "Hits")]
y_train <- Hitters[31:nrow(Hitters), c("Salary")]
xgb <- xgboost(as.matrix(X_train), as.matrix(y_train), nrounds = 200,verbose = 0)
```

5 The RMSE of the predictions

```r
y_pred <- predict(xgb, as.matrix(X_test))
# RMSE
```

```
rmse( y_test, y_pred)
```

```
## [1] 227.5978
```

6

	random forest	xgboost
RMSE	237.68	227.60

Comparing the RMSE value, xgboost gave us slightly better result.