BUS 41204 Review Session 3

Random Forest and Boosting

Chaoxing Dai
chaoxingdai@uchicago.edu

01/20/2018

Plan

- Random Forest using R (Regression)
- Boosting using R (Regression)
- Simulation (Classification)

Note: This would be hands-on session, run the rmd file on Rstudio instead of printing/reading the pdf.

Packages

```
#List the packages we need, install if missing, then load all of them
PackageList =c('MASS','gbm','tree','randomForest','rpart')
NewPackages=PackageList[!(PackageList %in%
                            installed.packages()[,"Package"])]
if(length(NewPackages)) install.packages(NewPackages)

lapply(PackageList,require,character.only=TRUE)#array function

set.seed(2018) #Always set the seed for reproducibility
```

Utility Function to Measure Performance

```
#Start stop watch timer
tic <- function(gcFirst = TRUE, type=c("elapsed", "user.self", "sys.self")){
   type <- match.arg(type)
   assign(".type", type, envir=baseenv())
   if(gcFirst) gc(FALSE)
   tic <- proc.time()[type]
   assign(".tic", tic, envir=baseenv())
   invisible(tic)
}

#Read elapsed time from stopwatch
toc <- function(){
   type <- get(".type", envir=baseenv())
   toc <- proc.time()[type]
   tic <- get(".tic", envir=baseenv())
   print(toc - tic)
   invisible(toc)
}
```

Getting/Splittin Data

```
download.file("https://raw.githubusercontent.com/ChicagoBoothML/MLClassData/mas
used_cars=read.csv(file="UsedCars_small.csv",head=TRUE,sep=",")

#check variables
sapply(used_cars, class)

N=dim(used_cars)[1]
p=dim(used_cars)[2]-1  #One of the columns is response, price
train_indices = sample(N, size = N * 0.75, replace = FALSE) #random partition

used_cars_train <- used_cars[train_indices, ]
used_cars_test <- used_cars[-train_indices, ]
hist(used_cars$price)
hist(used_cars_train$price)
hist(used_cars_test$price)
#Check if response distribution is the same
```

Random Forest: Input Arguments

```
tic()
frf = randomForest(price~.,                  #regression model
                   data=used_cars_train, #data set
                   mtry=3,      #number of variables to sample
                   ntree=500,   #number of trees to grow
                   nodesize=10,#minimum node size on trees (optional)
                   maxnodes=10,#maximum number of terminal nodes (optional)
                   importance=TRUE#calculate variable importance measure (optio
                   )
toc()
```

Random Forest: Returned Values

```
#Returned predictor
attributes(frf)

#Making Prediction
priceHat=predict(frf,newdata=used_cars_test)
cat('OOS RMSE=',sqrt(mean((used_cars_test$price-priceHat)^2)),'\n')

#Check the marginal fit
partialPlot(frf,used_cars_train,x.var=mileage,col = "red", lwd = 2)
points(used_cars_train$mileage, used_cars_train$price, xlab = "mileage", ylab =

partialPlot(frf,used_cars_train,x.var=year,col = "red", lwd = 2)
points(used_cars_train$year, used_cars_train$price, xlab = "year", ylab = "pric

#Checking Important Variables
varImpPlot(frf)
```

Random Forest vs Bagging

```
mtryv = c(p,sqrt(p))
ntreev = c(500,1000)
setrf = expand.grid(mtryv,ntreev)
colnames(setrf)=c("mtry","ntree")
RMSE_Train=rep(0,nrow(setrf))
RMSE_Test=rep(0,nrow(setrf))
###fit rf
for(i in 1:nrow(setrf)) {
   tic()
   cat("on randomForest fit ",i,", mtry=",setrf[i,1],", B=",setrf[i,2],"\n")
   frf = randomForest(price~.,data=used_cars_train,mtry=setrf[i,1],ntree=setrf[
   plot(frf,log="y")
   RMSE_Train[i]=sqrt(mean(frf$mse))#sqrt(mean((used_cars_train$price-frf$predic
   priceHat=predict(frf,newdata=used_cars_test)
   RMSE_Test[i]=sqrt(mean((used_cars_test$price-priceHat)^2))
   toc()
}
#(setrf)
print(RMSE_Train)
print(RMSE_Test)
```

Boosting: Input Arguments

Regression Setting:

```
fboost=gbm(price~.,                  #regression model
           data=used_cars_train, #data set
           distribution="gaussian",# boost the squared error, "tdist", 'laplace
           n.trees=500,              #Total number of trees/iterations
           interaction.depth = 1, #1 means additive, 2 means 2-way interaction,
           shrinkage=0.02            #Shrinkage parameter, weak predictor
           )
```

Boosting: Returned Values

```r
#Returned predictor
attributes(fboost)

#Making prediction
priceHat=predict(fboost,newdata=used_cars_test,n.trees=500)

cat('OOS RMSE=',sqrt(mean((used_cars_test$price-priceHat)^2)),'\n')

#Check the marginal fit
plot(fboost,i.var=3,col = "red", lwd = 2)
points(used_cars_train$mileage, used_cars_train$price, xlab = "mileage", ylab =

plot(fboost,i.var=4,col = "red", lwd = 2)
points(used_cars_train$year, used_cars_train$price, xlab = "year", ylab = "pric

# Out of Bag error estimation
plot(1:length(fboost$oobag.improve),fboost$oobag.improve, xlab = "iteration", y
```

Boosting: 2D Experiment

Data generating process:

```
n = 1000
X = matrix(runif(2*n, -1, 1), nrow = n)# uniform random number
Y = as.integer(rowSums(X * X) > 0.4)    # 0-1 response variable,

df = data.frame(X, Y)

plot(df$X1, df$X2, pch=c(3,1)[df$Y+1], col=c("red","blue")[df$Y+1])
#pch specifies plot symbols, "o" "+"
```

Boosting: Input Argument

Binary Outcome Setting:

```
fboost_2D=gbm(Y~.,                  #regression model
              data=df,              #data set
              distribution="adaboost",# "bernoulli" for logistic, "multinomial" for
              n.trees=5000,          #Total number of trees/iterations
              interaction.depth = 1, #1 means additive, 2 means 2-way interaction,
              shrinkage=0.01         #Shrinkage parameter, weak predictor
              )

plot(fboost_2D$train.error)
```

Boosting: Animation

```
MAX_TREES = 5000
GRID_SIZE = 20
x1 = seq(-1,1,length.out = GRID_SIZE)
x2 = seq(-1,1,length.out = GRID_SIZE)
z = rep(0, GRID_SIZE*GRID_SIZE)

DispSeq=c(1,seq(200,MAX_TREES, by = 200))

for (ntree in DispSeq) {
  for (i in 1:GRID_SIZE) {
    for (j in 1:GRID_SIZE) {
      z[(i-1)*GRID_SIZE+j] = predict(fboost_2D, data.frame(X1=x1[i],X2=x2[j]), n
      #log of odds returned
    }
  }
  image(x1, x2, matrix(z, nrow=GRID_SIZE), main=paste("Boosted trees = ", ntree
  points(df$X1, df$X2, pch=c(3,1)[df$Y+1], col=c("black","black")[df$Y+1])
}
```

Random Forest:

```
frf_2D = randomForest(as.factor(Y)~., #Turn into factor, classification activate
                      data=df, #data set
                      mtry=2,      #number of variables to sample
                      ntree=5000,  #number of trees to grow
                      importance=TRUE#calculate variable importance measure (optio
                      )

  for (i in 1:GRID_SIZE) {
    for (j in 1:GRID_SIZE) {
      z[(i-1)*GRID_SIZE+j] = predict(frf_2D, data.frame(X1=x1[i],X2=x2[j]))
      #log of odds returned
    }
  }
image(x1, x2, matrix(z, nrow=GRID_SIZE), main=paste("Random Forest (Final Vote)
points(df$X1, df$X2, pch=c(3,1)[df$Y+1], col=c("black","black")[df$Y+1])
```

References

Package manuals:

https://cran.r-project.org/web/packages/randomForest/randomForest.pdf
https://cran.r-project.org/web/packages/gbm/gbm.pdf

Making plots:

http://www.statmethods.net/advgraphs/parameters.html