**R: Bagging, Random Forest, and Boosting Tutorial**

*Goal:* Use decision trees in the bagging, random forest, and boosting techniques.

*Data:* Hitters dataset from the ISLR package for regression, Heart dataset for classification.

**Table of Contents**

1 --- Regression

1 ---- Bagged Tree

2 ---- Random Forest

6 ---- Boosted Tree

9 --- Classification

9 --- Bagged Tree

10 --- Random Forest

13 --- Boosted Tree

**Bagged Tree Regression**

First, let us load the necessary packages and data. We will start with a regression on the Hitters dataset from ISLR, using the Salary variable as our target, and split the data into training and testing as usual.

library(randomForest)

library(gbm)

library(ISLR)

# Set random seed

set.seed(1)

# Train Test Split

data(Hitters)

Hitters <- na.omit(Hitters) #Remove NA for demo

Hitters$Salary <- log(Hitters$Salary)

# Split training and testing

hitter.split <- sample(1:nrow(Hitters), size=nrow(Hitters) \* 0.7)

h.train <- Hitters[hitter.split,]

h.test <- Hitters[-hitter.split,]

Bagging (bootstrap aggregating) involves taking repeated samples from a single training dataset and training the model on the bootstrapped samples, and then averaging predictions. In this case, we use decision trees as our model to be bagged.

To implement in R, we use the randomForest package. Bagging is equivalent to a random Forest when the mtry parameter is set to the number of dimensions (p). Below, we test the bagging model with different numbers of trees. As the number of boostrapped samples increases, we see the mean-squared error decreases.

# Run the model and plot error against number of trees

bag.reg <- randomForest(Salary~., data=h.train, mtry=ncol(h.train) - 1, importance=TRUE, xtest=h.test[,-19], ytest=h.test$Salary)

plot(bag.reg)

Chart

Description automatically generated

**Random Forest Regression**

Similar to bagging, a Random Forest model also builds decision trees on bootstrapped samples, but it also samples a random subset of variables as well to decorrelate the trees.

Let us train a Random Forest model to predict salary. By randomly sampling variables, we can see which variables yield the largest changes in MSE and purity of the decision tree models. This allows us to plot variable importance, as well as the error over different number of tree samples as before.

rf.reg <- randomForest(Salary~., data=h.train, mtry=round(sqrt(ncol(h.train) - 1)), importance=TRUE, xtest=h.test[,-19], ytest=h.test$Salary)

importance(rf.reg)

%IncMSE IncNodePurity

AtBat 6.8208589 5.3013468

Hits 7.2111007 5.5410319

HmRun 3.3402870 2.0966444

Runs 7.1938548 3.9470050

RBI 4.3923829 4.2149623

Walks 9.6585636 5.8122108

Years 9.3257783 7.0807283

CAtBat 14.2825989 22.0486544

CHits 14.2774045 19.6344233

CHmRun 7.3880738 8.3821205

CRuns 12.9759520 19.3943834

CRBI 10.9668281 13.7673232

CWalks 10.4465485 15.0323036

League 2.0070706 0.2326236

Division 1.5433475 0.3374726

PutOuts 3.8334143 2.0528276

Assists -0.8453051 1.2217319

Errors 1.0191279 1.0067476

NewLeague 0.7084761 0.2270891

varImpPlot(rf.reg, cex=0.5)

Chart, scatter chart

Description automatically generated

plot(rf.reg)

Chart

Description automatically generated

From the plot, we estimate about 225 trees to be an acceptable number. Now, we must optimize the number of variables to sample for each tree, represented by the **mtry** parameter. Let us plot the out-of-bag error (error on observations not sampled in bagging) and the test error for different numbers of variables.

# Test different out of bag error

p <- ncol(h.train) - 1

oob.error <- double(p) #initialize empty vector

test.error <- double(p)

for(m in 1:p) {

fit <- randomForest(Salary ~ ., data=h.train, mtry=m, ntree=225)

oob.error[m] <- fit$mse[225]

test.error[m] <- mean((h.test$Salary - predict(fit, newdata=h.test))^2)

}

matplot(1:p, cbind(oob.error, test.error), pch=19, col=c("red", "blue"), type="b", ylab="Mean Squared Error" , xlab = "Number of Features")

legend("bottomright", c('OOB', 'Test') ,col=seq\_len(2),cex=0.8,fill=c("red", "blue"))

Chart, line chart

Description automatically generated

Observing the sampled variable count to be optimized at about 5, we compare normal bagging to the random forest and observe that the random forest performs best on the test set, especially at lower tree counts.

rf.reg <- randomForest(Salary~., data=h.train, mtry=4, importance=TRUE, xtest=h.test[,-19], ytest=h.test$Salary)

bag.err <- cbind(bag.reg$mse, bag.reg$test$mse, rf.reg$mse, rf.reg$test$mse)

colnames(bag.err) <- c("Bagging - OOB", "Bagging - Test", "RF - OOB", "RF - Test")

matplot(1:bag.reg$ntree, bag.err, type = "l", xlab="Number of Trees", ylab="MSE", col = c("red", "blue", "green", "black"))

legend("right", c("Bagging - OOB", "Bagging - Test", "Random Forest - OOB", "Random Forest - Test") ,col=seq\_len(2),cex=0.8,fill=c("red", "blue", "green", "black"))

A picture containing graphical user interface

Description automatically generated

# Calculate the MSE for training and testing using the best training

bag.train.mse <- min(bag.reg$mse)

bag.train.mse

[1] 0.2049195

bag.test.mse <- bag.reg$test$mse[which.min(bag.reg$mse)]

bag.test.mse

[1] 0.2335429

rf.train.mse <- min(rf.reg$mse)

rf.train.mse

[1] 0.1972515

rf.test.mse <- rf.reg$test$mse[which.min(rf.reg$mse)]

rf.test.mse

[1] 0.2125058

**Boosted Tree Regression**

Boosting is similar to bagging, but the sample dataset is modified depending on the results of the previously-grown tree. Let us fit a boosting model with sample parameters. Like the random forest, boosting can tell us how important each variable is to the predictive performance of the model, which we examine below.

reg.boost = gbm(Salary ~ ., data = h.train, n.trees = 5000, distribution = "gaussian", shrinkage = 0.01)

summary(reg.boost)

var rel.inf

CAtBat CAtBat 13.5702763

CWalks CWalks 10.3091036

CRBI CRBI 9.7192825

CRuns CRuns 8.5567722

Years Years 7.4510309

CHits CHits 6.3805556

PutOuts PutOuts 6.2466331

Walks Walks 5.8044810

Hits Hits 5.2351784

RBI RBI 4.9622496

CHmRun CHmRun 4.7397792

Assists Assists 3.5333664

Runs Runs 3.1134996

HmRun HmRun 2.6500261

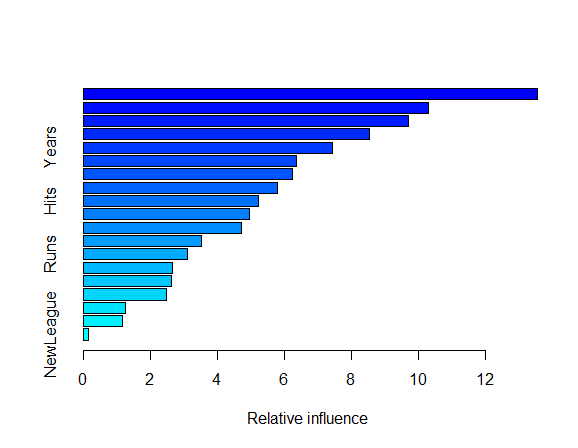
AtBat AtBat 2.6393074

Errors Errors 2.4892667

Division Division 1.2627198

League League 1.1829819

NewLeague NewLeague 0.1534896



plot(reg.boost, i="Walks")

Chart, histogram

Description automatically generated

plot(reg.boost, i="CHits")

Chart, line chart

Description automatically generated

plot(reg.boost, i="CRBI")

Chart, line chart

Description automatically generated

plot(reg.boost, i="Years")

Chart, line chart

Description automatically generated

train.mse <- min(reg.boost$train.error)

train.mse

[1] 0.06775375

test.mse <- mean((h.test$Salary - predict(reg.boost, h.test[,-19]))^2)

test.mse

[1] 0.2596795

Comparison of the three methods on the Hitters dataset:

|  |  |  |
| --- | --- | --- |
|  | Training MSE | Testing MSE |
| Bagging | 0.205 | 0.234 |
| Random Forest | 0.197 | 0.213 |
| Boosting | 0.068 | 0.260 |

**Bagged Tree Classification**

Each of the techniques discussed above can also be applied to classification. Let us now apply bagging to the Heart data used previously. Here our target variable is AHD, which is a binary (0/1) variable.

Again, bagging is equivalent to a random Forest when the mtry parameter is set to the number of dimensions (p). As the number of boostrapped samples increases, we see the mean-squared error decreases for the total OOB error and individual classes.

Heart <- read.csv('https://www.statlearning.com/s/Heart.csv')

Heart <- na.omit(Heart) #Remove NA for demo

Heart$AHD=factor(Heart$AHD)

heart.split <- sample(1:nrow(Heart), size=nrow(Heart) \* 0.7)

heart.train <- Heart[heart.split,]

heart.test <- Heart[-heart.split,]

bag.class <- randomForest(AHD~., data=heart.train, mtry=ncol(heart.train)-1, importance=TRUE, xtest=heart.test[,-15], ytest=factor(heart.test$AHD))

# Plot error of bagging and RF

bag.err <- cbind(bag.class$err[,1], bag.class$test$err[,1])

colnames(bag.err) <- c("Bagging - OOB", "Bagging - Test")

matplot(1:bag.reg$ntree, bag.err, type = "l", xlab="Number of Trees", ylab="Error", col = c("red", "blue"))

legend("topright", c("Bagging - OOB", "Bagging – Test"),col=seq\_len(2),cex=0.8,fill=c("red", "blue"))

Chart

Description automatically generated

**Random Forest Classification**

We again repeat the random forest model and compare to bagging for the Heart data classification problem. Similar to before, we examine the variable importance and determine the optimal number of features to be 4 based on the optimal classification error at that parameter value. The random forest again achieves lower classification error on the test data set than simple bagging.

rf.class <- randomForest(AHD~., data=heart.train,

mtry=round(sqrt(ncol(heart.train) - 1)), importance=TRUE)

importance(rf.class)

No Yes MeanDecreaseAccuracy MeanDecreaseGini

X -1.55162027 -1.734297121 -2.3406855 7.474078

Age 8.10343335 3.150167600 8.1626787 9.453801

Sex 10.70288620 6.594854083 12.0764978 3.974560

ChestPain 12.04032291 13.475451560 16.5741665 11.030015

RestBP 2.73683001 0.008168994 2.0873274 7.079586

Chol -3.42448702 -0.094344658 -2.4985396 6.730030

Fbs 2.70852925 -1.093711722 1.4157067 1.024329

RestECG -0.01644144 -1.335899728 -0.9872822 1.619458

MaxHR 8.71948479 3.760872119 8.9559679 10.871799

ExAng 5.12655269 6.340097086 7.7993532 6.355296

Oldpeak 14.09846982 11.257441529 17.8416187 12.547793

Slope 0.58512450 5.097300667 4.4079680 3.095427

Ca 19.44327372 10.909899067 20.2501541 10.528159

Thal 16.87702619 5.745933896 15.4097466 10.872181

varImpPlot(rf.class)

Chart, scatter chart

Description automatically generated

p <- ncol(heart.train) - 1

oob.error.class <- double(p) #initialize empty vector

for(m in 1:p) {

fit <- randomForest(AHD ~ ., data=heart.train, mtry=m, ntree=175)

conf.mat <- fit$err.rate[175]

oob.error.class[m] <- fit$err.rate[175, 'OOB']

}

matplot(1:p, oob.error.class, pch=19, col="red", type="b", ylab="Misclassification Error", xlab="mtry")

Chart, line chart

Description automatically generated

# Compare bagging and RF

# First get best RF

rf.class <- randomForest(AHD~., data=heart.train,

mtry=7, importance=TRUE, xtest=heart.test[,-15], ytest=factor(heart.test$AHD))

# Plot error of bagging and RF

bag.err <- cbind(bag.class$err[,1], bag.class$test$err[,1], rf.class$err[,1], rf.class$test$err[,1])

colnames(bag.err) <- c("Bagging - OOB", "Bagging - Test", "RF - OOB", "RF - Test")

matplot(1:bag.reg$ntree, bag.err, type = "l", xlab="Number of Trees", ylab="Error", col = c("red", "blue", "green", "black"))

legend("topright", c("Bagging - OOB", "Bagging - Test", "Random Forest - OOB", "Random Forest - Test") ,col=seq\_len(2),cex=0.8,fill=c("red", "blue", "green", "black"))

Chart, histogram

Description automatically generated

# Calculate best accuracy (1 - error) for bagging and random forest

bag.train.acc <- 1 - min(bag.class$err[,1])

bag.train.acc

[1] 0.8019324

bag.test.acc <- 1 - bag.class$test$err[which.min(bag.class$err[,1]),1]

bag.test.acc

Test

0.8111111

rf.train.acc <- 1 - min(rf.class$err[,1])

rf.train.acc

[1] 0.8019324

rf.test.acc <- 1 - rf.class$test$err[which.min(rf.class$err[,1]),1]

rf.test.acc

Test

0.8555556

**Boosted Tree Classification**

Finally, we repeat the same boosting process for classification as well using the Heart data. This again informs us of the variable importance at different values of our target. We plot some of the most important variables and how they relate to the target in our estimated model.

# format y for gbm, must be 0/1

heart.train$ChestPain = as.numeric(factor(heart.train$ChestPain))

heart.train$Thal = as.numeric(factor(heart.train$Thal))

AHD.0.1 <- ifelse(heart.train$AHD == 'Yes', 1, 0)

class.boost = gbm(AHD.0.1 ~ . - AHD, data = heart.train, n.trees = 5000, distribution = "adaboost", shrinkage = 0.01)

summary(class.boost)

var rel.inf

X X 13.8325910

Oldpeak Oldpeak 12.1708092

Age Age 10.6804303

MaxHR MaxHR 9.9390768

ChestPain ChestPain 9.4150735

Ca Ca 8.7407036

RestBP RestBP 7.9978014

Chol Chol 7.8754410

Thal Thal 6.5225347

Sex Sex 4.8179007

ExAng ExAng 4.1143576

Slope Slope 2.1584750

RestECG RestECG 1.3464411

Fbs Fbs 0.3883641

Chart, bar chart

Description automatically generated

plot(class.boost, i="ChestPain")

Chart, box and whisker chart

Description automatically generated

plot(class.boost, i="Chol")

Chart

Description automatically generated

plot(class.boost, i="X")

Chart, histogram

Description automatically generated

plot(class.boost, i="Ca")

Chart, histogram, box and whisker chart

Description automatically generated

# Calculate accuracy rates for training and testing

boost.train.acc <- 1 - min(class.boost$train.error)

boost.train.acc

[1] 0.7042179

AHD.test.0.1 <- ifelse(heart.test$AHD == 'Yes', 1, 0)

boost.test.acc <- 1 - sum(AHD.test.0.1 == ifelse(predict(class.boost, heart.test[,-15], type="response") > 0.5, 1, 0)) / nrow(heart.test)

boost.test.acc

[1] 0.7777778

Comparison of the three methods on the Heart dataset. For bagging and boosting we use the accuracies from the best plot.

|  |  |  |
| --- | --- | --- |
|  | Minimum Training Error | Testing Error |
| Bagging | 0.802 | 0.811 |
| Random Forest | 0.802 | 0.833 |
| Boosting | 0.704 | 0.778 |

Appendix Coding for the above instruction file

library(randomForest)

library(gbm)

library(ISLR)

# Set random seed

set.seed(1)

# Train Test Split

data(Hitters)

Hitters <- na.omit(Hitters) #Remove NA for demo

Hitters$Salary <- log(Hitters$Salary)

# Split training and testing

hitter.split <- sample(1:nrow(Hitters), size=nrow(Hitters) \* 0.7)

h.train <- Hitters[hitter.split,]

h.test <- Hitters[-hitter.split,]

# Run the model and plot error against number of trees

bag.reg <- randomForest(Salary~., data=h.train, mtry=ncol(h.train) - 1, importance=TRUE, xtest=h.test[,-19], ytest=h.test$Salary)

plot(bag.reg)

rf.reg <- randomForest(Salary~., data=h.train, mtry=round(sqrt(ncol(h.train) - 1)), importance=TRUE, xtest=h.test[,-19], ytest=h.test$Salary)

importance(rf.reg)

varImpPlot(rf.reg, cex=0.5)

plot(rf.reg)

# Test different out of bag error

p <- ncol(h.train) - 1

oob.error <- double(p) #initialize empty vector

test.error <- double(p)

for(m in 1:p) {

fit <- randomForest(Salary ~ ., data=h.train, mtry=m, ntree=225)

oob.error[m] <- fit$mse[225]

test.error[m] <- mean((h.test$Salary - predict(fit, newdata=h.test))^2)

}

matplot(1:p, cbind(oob.error, test.error), pch=19, col=c("red", "blue"), type="b", ylab="Mean Squared Error" , xlab = "Number of Features")

legend("bottomright", c('OOB', 'Test') ,col=seq\_len(2),cex=0.8,fill=c("red", "blue"))

rf.reg <- randomForest(Salary~., data=h.train, mtry=4, importance=TRUE, xtest=h.test[,-19], ytest=h.test$Salary)

bag.err <- cbind(bag.reg$mse, bag.reg$test$mse, rf.reg$mse, rf.reg$test$mse)

colnames(bag.err) <- c("Bagging - OOB", "Bagging - Test", "RF - OOB", "RF - Test")

matplot(1:bag.reg$ntree, bag.err, type = "l", xlab="Number of Trees", ylab="MSE", col = c("red", "blue", "green", "black"))

legend("right", c("Bagging - OOB", "Bagging - Test", "Random Forest - OOB", "Random Forest - Test") ,col=seq\_len(2),cex=0.8,fill=c("red", "blue", "green", "black"))

rf.reg <- randomForest(Salary~., data=h.train, mtry=4, importance=TRUE, xtest=h.test[,-19], ytest=h.test$Salary)

bag.err <- cbind(bag.reg$mse, bag.reg$test$mse, rf.reg$mse, rf.reg$test$mse)

colnames(bag.err) <- c("Bagging - OOB", "Bagging - Test", "RF - OOB", "RF - Test")

matplot(1:bag.reg$ntree, bag.err, type = "l", xlab="Number of Trees", ylab="MSE", col = c("red", "blue", "green", "black"))

legend("right", c("Bagging - OOB", "Bagging - Test", "Random Forest - OOB", "Random Forest - Test") ,col=seq\_len(2),cex=0.8,fill=c("red", "blue", "green", "black"))

# Calculate the MSE for training and testing using the best training mse

bag.train.mse <- min(bag.reg$mse)

bag.train.mse

bag.test.mse <- bag.reg$test$mse[which.min(bag.reg$mse)]

bag.test.mse

rf.train.mse <- min(rf.reg$mse)

rf.train.mse

rf.test.mse <- rf.reg$test$mse[which.min(rf.reg$mse)]

rf.test.mse

reg.boost = gbm(Salary ~ ., data = h.train, n.trees = 5000, distribution = "gaussian", shrinkage = 0.01)

summary(reg.boost)

plot(reg.boost, i="Walks")

plot(reg.boost, i="CHits")

plot(reg.boost, i="CRBI")

plot(reg.boost, i="Years")

boost.train.mse <- min(reg.boost$train.error)

boost.train.mse

boost.test.mse <- mean((h.test$Salary - predict(reg.boost, h.test[,-19]))^2)

boost.test.mse

### Heart Data ###

Heart <- read.csv('https://www.statlearning.com/s/Heart.csv')

Heart <- na.omit(Heart) #Remove NA for demo

Heart$AHD=factor(Heart$AHD)

heart.split <- sample(1:nrow(Heart), size=nrow(Heart) \* 0.7)

heart.train <- Heart[heart.split,]

heart.test <- Heart[-heart.split,]

bag.class <- randomForest(AHD~., data=heart.train, mtry=ncol(heart.train)-1, importance=TRUE, xtest=heart.test[,-15], ytest=factor(heart.test$AHD))

# Plot error of bagging and RF

bag.err <- cbind(bag.class$err[,1], bag.class$test$err[,1])

colnames(bag.err) <- c("Bagging - OOB", "Bagging - Test")

matplot(1:bag.reg$ntree, bag.err, type = "l", xlab="Number of Trees", ylab="Error", col = c("red", "blue"))

legend("topright", c("Bagging - OOB", "Bagging - Test"),col=seq\_len(2),cex=0.8,fill=c("red", "blue"))

# Random Forest

rf.class <- randomForest(AHD~., data=heart.train,

mtry=round(sqrt(ncol(heart.train) - 1)), importance=TRUE)

importance(rf.class)

varImpPlot(rf.class)

p <- ncol(heart.train) - 1

oob.error.class <- double(p) #initialize empty vector

for(m in 1:p) {

fit <- randomForest(AHD ~ ., data=heart.train, mtry=m, ntree=175)

conf.mat <- fit$err.rate[175]

oob.error.class[m] <- fit$err.rate[175, 'OOB']

}

matplot(1:p, oob.error.class, pch=19, col="red", type="b", ylab="Misclassification Error", xlab="Number of Features")

# Compare bagging and RF

# First get best RF

rf.class <- randomForest(AHD~., data=heart.train,

mtry=4, importance=TRUE, xtest=heart.test[,-15], ytest=factor(heart.test$AHD))

# Plot error of bagging and RF

bag.err <- cbind(bag.class$err[,1], bag.class$test$err[,1], rf.class$err[,1], rf.class$test$err[,1])

colnames(bag.err) <- c("Bagging - OOB", "Bagging - Test", "RF - OOB", "RF - Test")

matplot(1:bag.reg$ntree, bag.err, type = "l", xlab="Number of Trees", ylab="Error", col = c("red", "blue", "green", "black"))

legend("topright", c("Bagging - OOB", "Bagging - Test", "Random Forest - OOB", "Random Forest - Test") ,col=seq\_len(2),cex=0.8,fill=c("red", "blue", "green", "black"))

# Calculate best accuracy (1 - error)

bag.train.acc <- 1 - min(bag.class$err[,1])

bag.train.acc

bag.test.acc <- 1 - bag.class$test$err[which.min(bag.class$err[,1]),1]

bag.test.acc

rf.train.acc <- 1 - min(rf.class$err[,1])

rf.train.acc

rf.test.acc <- 1 - rf.class$test$err[which.min(rf.class$err[,1]),1]

rf.test.acc

# format y for gbm, must be 0/1

heart.train$ChestPain = as.numeric(factor(heart.train$ChestPain))

heart.train$Thal = as.numeric(factor(heart.train$Thal))

AHD.0.1 <- ifelse(heart.train$AHD == 'Yes', 1, 0)

class.boost = gbm(AHD.0.1 ~ . - AHD, data = heart.train, n.trees = 5000, distribution = "adaboost", shrinkage = 0.01)

summary(class.boost)

plot(class.boost, i="ChestPain")

plot(class.boost, i="Chol")

plot(class.boost, i="X")

plot(class.boost, i="Ca")

# Calculate error rates for training and testing

boost.train.err <- 1 - min(class.boost$train.error)

boost.train.err

AHD.test.0.1 <- ifelse(heart.test$AHD == 'Yes', 1, 0)

boost.test.err <- sum(AHD.test.0.1 == ifelse(predict(class.boost, heart.test[,-15], type="response") > 0.5, 1, 0)) / nrow(heart.test)

boost.test.err