DAV_hw04

Weijun Zhu

November 19, 2019

Contents

Problem 1 (30 points)

Generate a simulated data set with 20 observations in each of three classes (i.e. 60 observations total), and 50 variables. Be sure to add a mean shift to the observations in each class so that there are three distinct classes.

Hint: Use rnorm() function in R.

```
data <- rbind(matrix(rnorm(20*50, mean = 0), nrow = 20),
              matrix(rnorm(20*50, mean=0.7), nrow = 20),
              matrix(rnorm(20*50, mean=1.4), nrow = 20))
dim(data)
```

```
## [1] 60 50
```

A.

Perform K-means clustering of the observations with K = 3. How well do the clusters that you obtained in K-means clustering compare to the true class labels?

Hint: You can use the table() function in R to compare the true class labels to the class labels obtained by clustering. Be careful how you interpret the results: K-means clustering will arbitrarily number the clusters, so you cannot simply check whether the true class labels and clustering labels are the same.

```
model_km <- kmeans(data, centers=3)
true_class = c(rep(1,20), rep(2,20), rep(3,20))
table(model_km$cluster, true_class)
```

```
##    true_class
##      1  2  3
##   1 20  0  0
##   2  0 20  0
##   3  0  0 20
```

Conclusion: Every cluters separate perfectly!

B.

Perform K-means clustering with K = 2. Describe your results.

```
model_km <- kmeans(data, centers=2)
true_class = c(rep(1,20), rep(2,20), rep(3,20))
table(model_km$cluster, true_class)
```

```
##    true_class
##      1  2  3
##   1  0 18 20
##   2 20  2  0
```

Conclusion: The consequence looks not good.

C.

Now perform K-means clustering with K = 4, and describe your results

```
model_km <- kmeans(data, centers=4)
true_class = c(rep(1,20), rep(2,20), rep(3,20))
table(model_km$cluster, true_class)
```

```
##    true_class
##      1  2  3
##   1  0 17  0
##   2  0  0 15
##   3 20  0  0
##   4  0  3  5
```

Conclusion: The consequence of this time looks so bad.

Problem 2. (50 points)

Using the Carseats dataset in ISLR package, predict Sales using regression trees and related approaches, treating the response as a quantitative variable.

```
library(ISLR)
data(Carseats)
str(Carseats)
```

```
## 'data.frame':    400 obs. of  11 variables:
##  $ Sales      : num  9.5 11.22 10.06 7.4 4.15 ...
##  $ CompPrice  : num  138 111 113 117 141 124 115 136 132 132 ...
##  $ Income     : num  73 48 35 100 64 113 105 81 110 113 ...
##  $ Advertising: num  11 16 10 4 3 13 0 15 0 0 ...
##  $ Population : num  276 260 269 466 340 501 45 425 108 131 ...
##  $ Price      : num  120 83 80 97 128 72 108 120 124 124 ...
##  $ ShelveLoc  : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...
##  $ Age        : num  42 65 59 55 38 78 71 67 76 76 ...
##  $ Education  : num  17 10 12 14 13 16 15 10 10 17 ...
##  $ Urban      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
##  $ US         : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
```

A.

Split the data set into a training set and a test set.

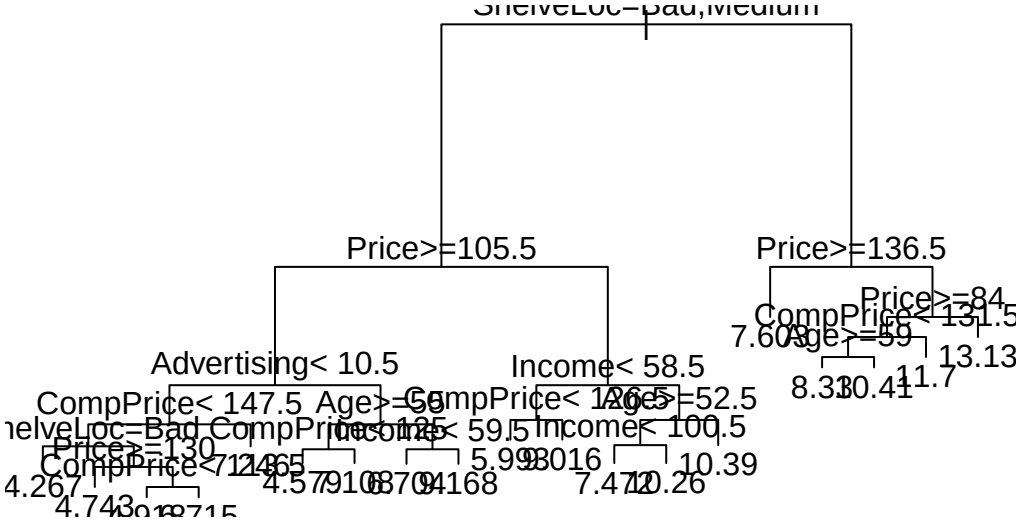
```
split <- sample(nrow(Carseats), size=0.7*nrow(Carseats))
training <- Carseats[split,]
testing <- Carseats[-split,]
```

B.

Fit a regression tree to the training set. Plot the tree, and interpret the results. What test error rate do you obtain?

```
library(rpart)
set.seed(9)
rpart_model <- rpart(Sales ~ ., data = training, method = 'anova')
```

```
plot(rpart_model)
text(rpart_model, pretty = 0)
```




```
tree_df_pred <- predict(rpart_model, testing)
mean((testing$Sales - tree_df_pred)^2)
```

```
## [1] 4.526473
```

C.

Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test error rate?

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
p <- dim(Carseats)[2]-1
bagging <- randomForest(Sales~., data = training, importance = T, mtry = p, ntree = 500)
bagging_pred <- predict(bagging, testing)
importance(bagging)
```

```
##                %IncMSE IncNodePurity
## CompPrice   29.14164435    231.268857
```

```
## Income      16.73265520    150.016127
## Advertising 21.44775461    168.139974
## Population  -1.20101509     72.412589
## Price       61.28107875    584.880324
## ShelveLoc   73.55833231    634.008642
## Age         21.39576555    216.250149
## Education    0.93955151     61.844909
## Urban       -1.00131012      8.512551
## US          -0.04151209      7.528448
```

```
mean((bagging_pred-testing$Sales)^2)
```

```
## [1] 2.66005
```

Conclusion: Yes, this tree improves the test error rate.

E.

Use random forests to analyze this data. What test error rate do you obtain? Use the importance() function to determine which variables are most important. Describe the effect of m, the number of variables considered at each split, on the error rate obtained.

```
MSE_list <- sapply(1:(p-1), function(i){
  randomF <- randomForest(Sales~., data = training, importance = T, mtry = i, ntree = 500)
  randomF_pred <- predict(randomF, testing)
  mean((randomF_pred-testing$Sales)^2)
})
minNbrTrees <- which.min(MSE_list)
min(MSE_list)
```

```
## [1] 2.61999
```

```
randomF <- randomForest(Sales~., data = training, importance = T, mtry = minNbrTrees, ntree = 500)
importance(randomF)
```

```
##                %IncMSE IncNodePurity
## CompPrice   24.7993516    216.808653
## Income      14.8897980    161.760087
## Advertising 19.7761916    175.907992
## Population  -1.3674236     80.252040
## Price       52.6762933    566.285043
## ShelveLoc   72.2493093    632.825503
## Age         22.3923659    231.754529
## Education    2.2128289     64.562757
## Urban       -1.2347254     10.373404
## US           0.2668537      9.759704
```

Problem 3 (20 points)

In this problem, using the Auto dataset in ISLR package, apply support vector approaches to predict whether a given car gets high or low gas mileage.

```r
data(Auto)
str(Auto)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 1
```

A.

Create a binary variable that takes on a 1 for cars with gas mileage above the median, and a 0 for cars with gas mileage below the median.

```r
library(e1071)
library(plyr)
set.seed(9)
med <- median(Auto$mpg)
df <- mutate(Auto, mpgF = as.factor(ifelse(mpg > med,1,0)))
N <- dim(Auto)[1]
trainSample <- sample(1:N, N/2)
train <- df[trainSample,]
test <- df[-trainSample,]
```

B.

Fit a support vector classifier to the data with various values of cost, in order to predict whether a car gets high or low gas mileage. Report the cross-validation errors associated with different values of this parameter. Comment on your results.

```r
svm.tune <- tune(svm, mpgF ~ ., data = train, kernel = "linear", ranges = list(cost = c(0.01, 0.1, 1, 5
summary(svm.tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     5
##
## - best performance: 0.03078947
##
## - Detailed performance results:
##    cost      error dispersion
```

```
## 1 1e-02 0.08684211 0.05388850
## 2 1e-01 0.05657895 0.03905726
## 3 1e+00 0.03605263 0.02490518
## 4 5e+00 0.03078947 0.02652113
## 5 1e+01 0.03078947 0.02652113
## 6 1e+02 0.03078947 0.02652113
```

Conclusion: The best result is when the cost=1, and we get a cross validation error of 0.02552632.