Stats 415 homework3 solution

April Cho

2/5/2018

problem 1

We first divide the data into training and test data using the first 90% of the observations for training and the remaining 10% for testing. Both training error and test error are slightly lower in model 1 but the difference is very small. Thus we can conclude that two models aren't too different from each other.

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.2.5
```

```
library(FNN)
train_id = 1:(nrow(Carseats)*0.9)
trainCarseats = Carseats[train_id,]
testCarseats = Carseats[-train_id,]

mod1 <- lm(Sales ~ ., data=trainCarseats)
mod2 <- lm(Sales ~ . -Population-Education-Urban-US, data=trainCarseats)
mse <- function(model, y, data) {
  yhat <- predict(model, data)
  mean((y - yhat)^2)
}

#for model 1
train_mse1 = mse(mod1, trainCarseats$Sales, trainCarseats)
test_mse1 = mse(mod1, testCarseats$Sales, testCarseats)

#for model 2
train_mse2 = mse(mod2, trainCarseats$Sales, trainCarseats)
test_mse2 = mse(mod2, testCarseats$Sales, testCarseats)

train_mse1; test_mse1;
```

```
## [1] 1.010792
```

```
## [1] 0.9903956
```

```
train_mse2; test_mse2;
```

```
## [1] 1.021013
```

```
## [1] 1.0041
```

problem 2

With K=1, the KNN model fits every data points in the training data(overfitting). Hence, we expect a better(=lower) training error with K = 1 than with K = 10. However, we expect a higher test error with K = 1 due to high variance. KNN with K=10 is slightly less complex model than KNN with K=1 so we would expect lower test error with K=10.

problem 3

```
summary(Carseats[c("CompPrice", "Income", "Advertising", "Price", "Age")])
```

```
##    CompPrice       Income        Advertising         Price
##  Min.   : 77   Min.   : 21.00   Min.   : 0.000   Min.   : 24.0
##  1st Qu.:115   1st Qu.: 42.75   1st Qu.: 0.000   1st Qu.:100.0
##  Median :125   Median : 69.00   Median : 5.000   Median :117.0
##  Mean   :125   Mean   : 68.66   Mean   : 6.635   Mean   :115.8
##  3rd Qu.:135   3rd Qu.: 91.00   3rd Qu.:12.000   3rd Qu.:131.0
##  Max.   :175   Max.   :120.00   Max.   :29.000   Max.   :191.0
##       Age
##  Min.   :25.00
##  1st Qu.:39.75
##  Median :54.50
##  Mean   :53.32
##  3rd Qu.:66.00
##  Max.   :80.00
```

From the summary of the predictors, we can check that ranges of the predictors are quite different. If one variable contains much larger numbers because of the units or range of the variable, it will dominate other variables in the distance measurements. But this doesn't necessarily mean that it should be such an important variable. Hence we would standardize the variables first.

problem 4

The KNN regression is fitted for different values of K. Notice that the training and test data(predictors only) are scaled inside the function to fit the KNN regression.

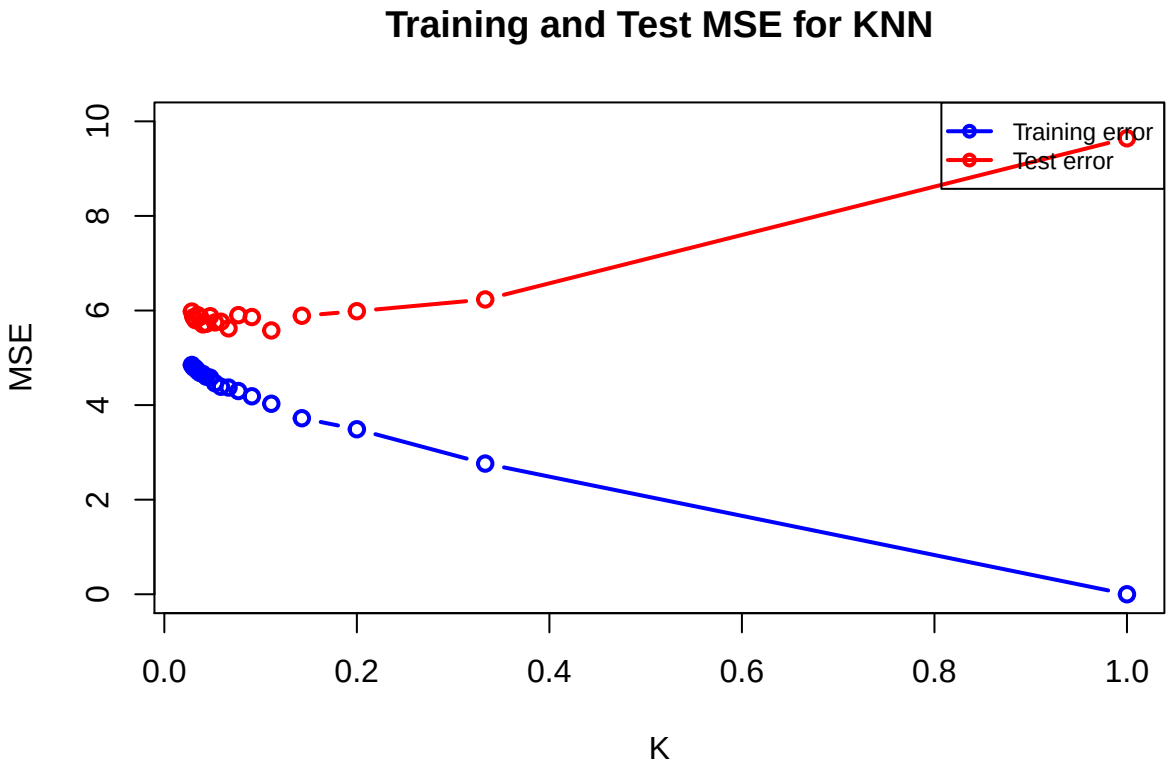
```
krange = seq(1, 35, by= 2)

trainMSE = c() #creating null vector
testMSE = c()

for(i in 1:length(krange)){
  knnTrain <- knn.reg(train = scale(trainCarseats[c("CompPrice", "Income", "Advertising", "Price", "Age
                      test = scale(trainCarseats[c("CompPrice", "Income", "Advertising", "Price", "Age"]
                      y = trainCarseats$Sales, k = krange[i])
  trainMSE[i] <- mean((trainCarseats$Sales - knnTrain$pred)^2)

  knnTest <- knn.reg(train = scale(trainCarseats[c("CompPrice", "Income", "Advertising", "Price", "Age"]
                     test = scale(testCarseats[c("CompPrice", "Income", "Advertising", "Price", "Age")]]
                     y = trainCarseats$Sales, k = krange[i])
  testMSE[i] <- mean((testCarseats$Sales - knnTest$pred)^2)
}

plot(I(1/krange), trainMSE, type = "b", lwd = 2, col = "blue", ylim=c(0,10),
     main = "Training and Test MSE for KNN", xlab = "K", ylab = "MSE")
lines(I(1/krange), testMSE, type = "b", lwd = 2, col = "red")
legend("topright", legend = c("Training error", "Test error"),
       col = c("blue", "red"), cex = .75,
       lwd = c(2, 2), pch = c(1, 1), lty = c(1, 1))
```




```
krange[which.min(trainMSE)]
```

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

```
krange[which.min(testMSE)]
```

```
## [1] 9
```

$K = 1$ achieves the lowest training error and $K = 9$ achieves the lowest test error. Training error is lowest at K = 1 and increases as K increases. Test error first decreases and then increases as K increases (U-shape function). It might be easier for you to check the shape when the x-axis is just values of K instead of inverse of K. For training error optimal K is 1, and for test error optimal K is 9.

problem 5

KNN regression model is fitted with the optimal K=9 chosen by test error in problem 4.
The $residuals = y_i - \hat{y}_i = oberservation_i - fitted.value_i$. Hence we calculate residuals for KNN regression and for model 2 by estimating $\hat{y}_i$ for test data and then subtracting it from y_i in test data. Notice that we fit the models on training data but test them on test data.

The points are quite randomly scattered around 0 but we can see some high residuals in KNN plot. The range of residuals is smaller for model 2 than in KNN model.

```
knnTest_9 <- knn.reg(train = scale(trainCarseats[,c(2:4,6,8)]),
                     test = scale(testCarseats[,c(2:4,6,8)]),
                     y = trainCarseats$Sales, k = 9)
mod2_residual = (testCarseats$Sales - predict(mod2,testCarseats))
knn_residual = (testCarseats$Sales - knnTest_9$pred)

par(mfrow=c(1,2))
plot(mod2_residual ~ predict(mod2, testCarseats), ylim=c(-5,7), xlim=c(1,16),
```

```
     main="Residuals plot(model2)", xlab="fitted values", ylab="residuals")
abline(h=0, col="blue")
plot(knn_residual ~ knnTest_9$pred, ylim=c(-5,7),xlim=c(1,16),
     main="Residuals plot(KNN)", xlab="fitted values", ylab="residuals")
abline(h=0, col="blue")
```