STAT542, Homework 1

Shuhui Guo

Problem 1

a)

Since $\hat{\Sigma}_{MLE} = \frac{1}{N}[(X - \hat{X})^T(X - \hat{X})]$, we can get the sample variance-covariance matrix $\hat{\Sigma} =$

```
X_hat <- colMeans(X)
Sigma_hat<-t(X-matrix(rep(X_hat,N),N,P,byrow = T))%*%(X-matrix(rep(X_hat,N),
           N,P,byrow = T))/N
Sigma_hat
```

```
##             [,1]      [,2]      [,3]       [,4]
## [1,] 0.97247606 0.3909800 0.2105751 0.04279479
## [2,] 0.39097995 0.9679975 0.4298725 0.11644234
## [3,] 0.21057509 0.4298725 1.0022939 0.38896065
## [4,] 0.04279479 0.1164423 0.3889606 0.92359611
```

And $\hat{\Sigma}^{-1/2} =$

```
"%^%" <- function(x, n)
  with(eigen(x), vectors %*% (values^n * t(vectors)))
Sigma_hat %^% (-0.5)
```

```
##              [,1]         [,2]        [,3]         [,4]
## [1,]  1.087195001 -0.214630433 -0.05304807  0.009189527
## [2,] -0.214630433  1.164390335 -0.24338412  0.006920835
## [3,] -0.053048066 -0.243384122  1.15864541 -0.237377262
## [4,]  0.009189527  0.006920835 -0.23737726  1.115351840
```

b)

Write the function:

```
mydist <- function(x1,x2) sqrt(sum((x1 - x2) ^ 2))
x = c(0.5,0.5,0.5,0.5)
save <- c()

for(i in 1:200){
save[i] <- mydist(X[i,],x)
}

order = order(save, decreasing = FALSE)[1:5]
estimation = mean(Y[order])
```

The row numbers of the closest 5 subjects are 188, 165, 144, 36, 154

The 5-NN estimation at the target point is 1.3466389

c)

Write a function:

```
mydist2 <- function(x1,x2,s) sqrt((x1 - x2)%*% solve(s) %*% (x1-x2))
x = c(0.5,0.5,0.5,0.5)
save2 <- c()

for(i in 1:200){
save2[i] <- mydist2(X[i,],x,Sigma_hat)
}

order2 = order(save2, decreasing = FALSE)[1:5]
estimation2 = mean(Y[order2])
```

The row numbers of the closest 5 subjects are 188, 49, 144, 165, 36

The 5-NN estimation at the target point is 0.5615577

d)

The 5-NN estimation based on the Euclidean distance seems better.

From the results of b) and c), we can see that four of the closest 5 subjects based on the Euclidean and Mahalanobis distance are the same. But the 5-NN estimations at the target point are quite different, which indicates that the different subject makes a big difference. This situation may happen by instance and perhaps the Mahalanobis distance is not suitable here.

In addition, the 5-NN estimation based on the Euclidean distance is closer to $Y = x * \beta = 1.5$, so it is better.

Problem 2

a)

The regression model

$$Y = f(X) + \epsilon \tag{1}$$

where $\mathrm{E}(\epsilon) = 0$ and $\mathrm{Var}(\epsilon) = \sigma^2$.

The 5-NN estimation

$$\hat{y} = \frac{1}{5} \sum_{x_i \in N_5(x)} y_i \tag{2}$$

where N_5(x) defines the 5 samples from the training data that are closest to x.

Since the degrees of freedom is defined as $\sum_{i=1}^{n} Cov(\hat{y}_i, y_i)/\sigma^2$, the degrees of freedom of the model using $k = 5$ is

$$\sum_{i=1}^{n} Cov(\hat{y}_i, y_i)/\sigma^2 = \sum_{i=1}^{n} \frac{1}{5} Cov(y_i, y_i)/\sigma^2 = \sum_{i=1}^{n} \frac{1}{5}\sigma^2/\sigma^2 = \frac{n}{5} \tag{3}$$

b)

Generate a design matrix X:

```
library(MASS)
set.seed(1)
P = 4
N = 200
V <- diag(P)
X = as.matrix(mvrnorm(N, mu=rep(0,P), Sigma=V))
```

Define the model $\overline{Y} = f(X) = X_1 + X_2 + 2X_3 + 2X_4$

```
beta = as.matrix(c(1, 1, 2, 2))
Y = X %*% beta + rnorm(N)  #generate the response variables by adding noise
```

```
library(kknn)
```

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

```
myfun <- function(times) {
  y <- matrix(0,times,N)
  yhat <- matrix(0,times,N)  #matrix of the estimations
  for (i in 1:times){
    Y = X %*% beta + rnorm(N)
    y[i,] = Y
    k = 5
    knn.fit = kknn(Y ~ X, train = data.frame(x = X, y = Y),
              test = data.frame(x = X), k = k, kernel = "rectangular")  #from intro.r
    yhat[i,] = knn.fit$fitted.values  #obtain the estimations
  }

  #Calculate the covariance
  ybar = X %*% beta
  cov <- c()
  for (i in 1:N){
  cov[i] = sum((yhat[,i]-mean(yhat[,i]))%*%(y[,i]-ybar[i]))/(times-1)
  }
  covariance = sum(cov)
  return(covariance)
}

myfun(20)
```

```
## [1] 42.45304
```

My estimated degrees of freedom is close to the theoretical value.

c)

$Cov(\hat{y}, y) = Cov(X(X^TX)^{-1}X^Ty, y) = X(X^TX)^{-1}X^TCov(y, y) = \sigma^2X(X^TX)^{-1}X^T$

Then, $df(\hat{f}) = \frac{1}{\sigma^2}Trace(Cov(\hat{y}, y)) = \frac{1}{\sigma^2}Trace(\sigma^2X(X^TX)^{-1}X^T) = Trace(X(X^TX)^{-1}X^T) = Trace((X^TX)^{-1}X^TX) = Trace(I_p) = p$

Therefore, the theoretical degrees of freedom for this linear regression is p.

Problem 3

```
library(ElemStatLearn)
```

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

```
library(class)
```

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

```
data(SAheart)
Y = SAheart$chd
X = cbind(SAheart$age,SAheart$tobacco)

nfold = 10
set.seed(1)
infold = sample(rep(1:nfold, length.out=length(Y)))
mydata = data.frame(x = X, y = Y)

K = 50  #maximum number of k that I am considering
errorMatrix = matrix(NA, K, nfold)  #save the prediction error of each fold
errorMatrix.train = matrix(NA, K, nfold)

for (l in 1:nfold)
{
    for (k in 1:K)
    {
        #calculate testing error
      knn.fit = knn(train = mydata[infold != l, 1:2],
                test = mydata[infold == l, 1:2],
                cl = mydata[infold != l,3], k = k, prob = FALSE)
        errorMatrix[k, l] = 1-mean(knn.fit == mydata$y[infold == l])
        #calculate training error
        knn.fit.train = knn(train = mydata[infold != l, 1:2],
                        test = mydata[infold != l, 1:2],
                        cl = mydata[infold != l,3], k = k, prob = FALSE)
        errorMatrix.train[k, l] = 1-mean(knn.fit.train == mydata$y[infold != l])
    }
}
best = which.min(apply(errorMatrix, 1, mean))
```

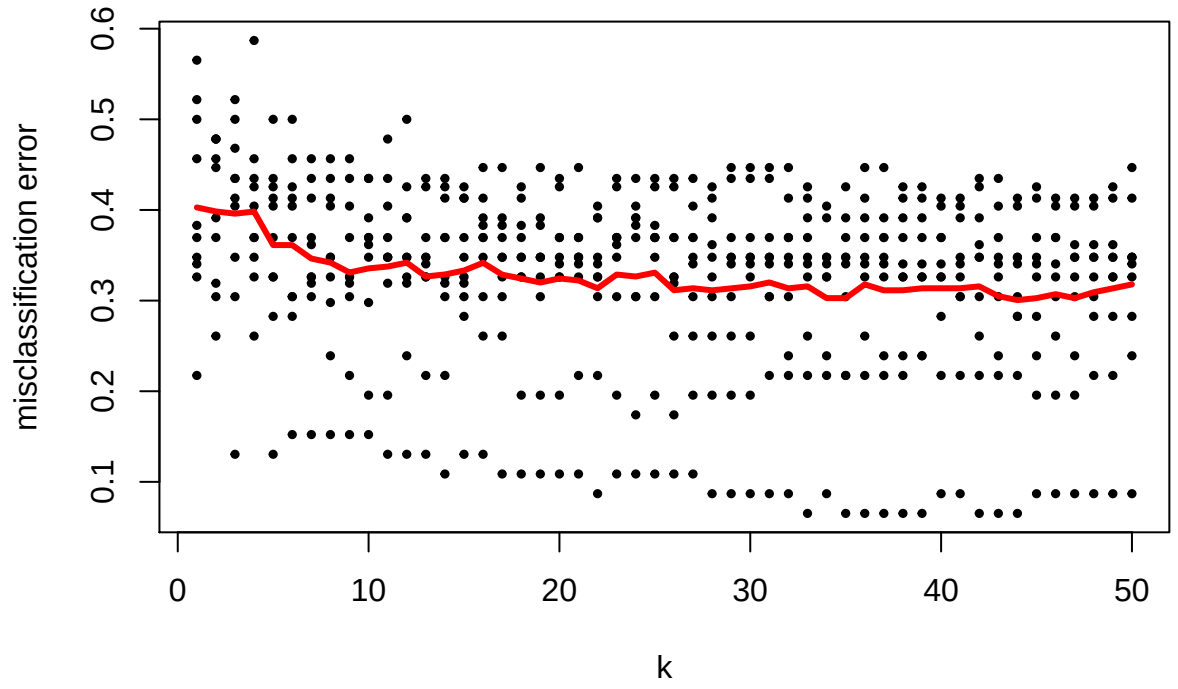
We can report the training error:

```
errorMatrix.train[1:5,1:5]
```

```
##             [,1]       [,2]       [,3]       [,4]       [,5]
## [1,] 0.03614458 0.02891566 0.03365385 0.02884615 0.02884615
## [2,] 0.19036145 0.19518072 0.19230769 0.17307692 0.20913462
## [3,] 0.20722892 0.22409639 0.20673077 0.18509615 0.21153846
## [4,] 0.26746988 0.25783133 0.24038462 0.25721154 0.26201923
## [5,] 0.26265060 0.27228916 0.24519231 0.24759615 0.26923077
```

And we plot the averaged cross-validation error curve for difference choices of k:

```
plot(rep(1:K, nfold), as.vector(errorMatrix), pch = 19, cex = 0.5,
     ylab="misclassification error",xlab="k")   #plot the results
points(1:K, apply(errorMatrix, 1, mean), col = "red", pch = 19, type = "l", lwd = 3)
```



The best k is 44.