STA545 HW2 Solution

Problem 1

We first construct a dataset for problem 1

```
obs <- 1:6
X1 <- c(0,2,0,0,-1,1)
X2 <- c(3,0,1,1,0,1)
X3 <- c(0, 0, 3, 2, 1, 1)
Y <- c("Red", "Red", "Red", "Green", "Green", "Red")
data1 <- data.frame(obs, X1, X2, X3, Y)
data1
```

```
##   obs X1 X2 X3     Y
## 1   1  0  3  0   Red
## 2   2  2  0  0   Red
## 3   3  0  1  3   Red
## 4   4  0  1  2 Green
## 5   5 -1  0  1 Green
## 6   6  1  1  1   Red
```

(a). Compute the Euclidean distance between each observation and the test point, X1=X2=X3=0.

By Euclidean distance formula:

$$d = \sqrt{\sum_{i=1}^{3}(x_i - x_i^*)^2}$$

where x_i^* is the i-th variable for the test data point (0, 0, 0), we can obtain the distances between the test data point and each training sample.

Then, we add a column "distance" into data1.

```
X.star <- c(0, 0, 0)
# distance^2
dist.sq <- apply((data1[, c("X1", "X2","X3")] - X.star)^2, 1, sum)
# distance
dist <- round(sqrt(dist.sq), 2)
# add "distance" column into data1
data1$dist <- dist
data1
```

```
##   obs X1 X2 X3     Y dist
## 1   1  0  3  0   Red 3.00
## 2   2  2  0  0   Red 2.00
## 3   3  0  1  3   Red 3.16
## 4   4  0  1  2 Green 2.24
## 5   5 -1  0  1 Green 1.41
## 6   6  1  1  1   Red 1.73
```

(b). What is our prediction with k = 1.

If the hyper-parameter in K-NN is equal to 1, then we just need to find the label of the nearest training data point.

```
index.1 <- which(dist==min(dist))
data1[index.1, ]
```

```
##   obs X1 X2 X3     Y dist
## 5   5 -1  0  1 Green 1.41
```

So our prediction based on K-NN with k=1 is "Green".

(c). What is our prediction with k = 3.

We first find the top 3 labels with the smallest distances:

```
sort.info <- sort(dist, index.return=TRUE)
index.2 <- sort.info$ix[1:3]
data1[index.2, ]
```

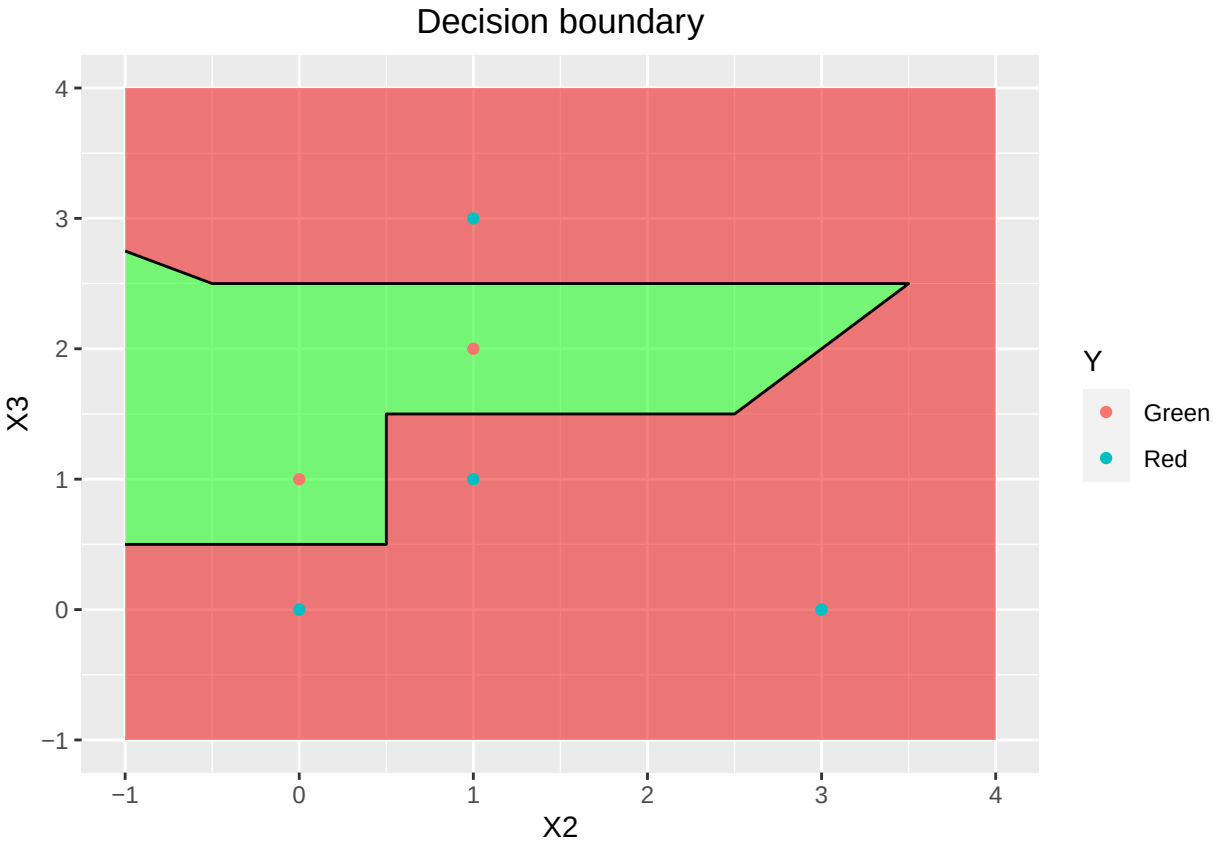
```
##   obs X1 X2 X3     Y dist
## 5   5 -1  0  1 Green 1.41
## 6   6  1  1  1   Red 1.73
## 2   2  2  0  0   Red 2.00
```

Then we can find there are 2 "Red" and 1 "Green". The prediction is "Red" if we use voting method (2>1).

(d). Suppose we only use features X2 and X3. Please draw the decision boundary of the k-nearest neighbor classifier with k = 1 in a figure.

We use the following ggplot2 package to draw th decision boundary.

```
library(ggplot2)
ggplot(data=data1) +
  geom_ribbon(aes(x=x, ymin=ymin, ymax=ymax), fill="red2", alpha=0.5,
  data=data.frame(x=c(-1, -0.5, 4), ymin=c(2.75, 2.5, 2.5), ymax=c(4,4, 4))) +
  geom_ribbon(aes(x=x, ymin=ymin, ymax=ymax), fill="green", alpha=0.5,
  data=data.frame(x=c(-1, -0.5, 0.5, 0.5, 2.5, 3.5),
  ymin=c(0.5, 0.5, 0.5, 1.5, 1.5, 2.5),
  ymax=c(2.75, 2.5, 2.5, 2.5, 2.5, 2.5))) +
  geom_ribbon(aes(x=x, ymin=ymin, ymax=ymax), fill="red2", alpha=0.5,
  data=data.frame(x=c(-1, 0.5, 0.5, 2.5, 3.5, 4),
  ymin=rep(-1, 6),
  ymax=c(0.5, 0.5, 1.5, 1.5, 2.5, 2.5))) +
  geom_path(aes(x=x, y=y), data=data.frame(x=c(-1, 0.5, 0.5, 2.5, 3.5, -0.5, -1),
  y=c(0.5, 0.5, 1.5, 1.5, 2.5, 2.5, 2.75))) +
  xlim(-1, 4) +
  ylim(-1, 4) +
  geom_point(aes(x=X2, y=X3, color=Y)) +
  labs(y="X3", x = "X2") +
  ggtitle("Decision boundary") +
  theme(plot.title = element_text(hjust = 0.5))
```



All 6 training data points are shown above, with their colors indicating their labels. The decision boundary is shown in black. Data points falling inside the black boundary (green shade) will be classified as "Green", and those outside the black boundary (red shade) will be classified as "Red". (Note that the decision boundaries at $X_3 = 0.5$ and $X_3 = 2.75$ on the left of the figure extend infinitely in the same direction).

(e). If the Bayes decision boundary in this problem is highly nonlinear, would we expect the best value for k to be large or small? Why?

We would expect the best value for k to be small because the k-NN method becomes more flexbible as k decreases.

Problem 2

(a).

We first load data into R and transform the labels into 1 or -1. Files "2.txt" and "3.txt" are extracted from the original training data with label "2" and "3", respectively.

(We can also load the original training dataset by "data <- read.table("train.txt")" and then extract data with label 2 and 3 by "train <- data[data$V1 == 2|data$V1 == 3,]").

```
# training data
train.2 <- read.table("2.txt",header = F, sep=",")
train.3 <- read.table("3.txt",header = F, sep=",")
train.2$Y = 1
train.3$Y = -1
train <- rbind(train.2, train.3)
#dim(train)
```

```
# testing data
test <- read.table("test.txt", header = F, sep=" ")
test.data <- test[test$V1 == 2 | test$V1 == 3, ]
test.data$Y = test.data$V1
test.data$Y[test.data$Y == 2] = 1
test.data$Y[test.data$Y == 3] = -1
testing = subset(test.data, select = -c(V1))
colnames(testing) = colnames(train)
#dim(test.data)
```

We can find that there are 1389 samples in training data and 364 samples in testing data.

For problem (a), we build a linear regression classifier, i.e.

$$\hat{f}(x) = \begin{cases} 1, if \hat{y} > 0 \\ -1, if \hat{y} \leq 0 \end{cases}$$

```
fit.linear <- lm(Y~., data=train)
fitted.linear <- predict(fit.linear, newdata = train)
fitted.linear[fitted.linear > 0 ] = 1
fitted.linear[fitted.linear <= 0 ] = -1
# train error
train.error.linear <- mean(fitted.linear != train$Y)
# test error
pred.linear <- predict(fit.linear, newdata = testing)
pred.linear[pred.linear > 0] = 1
pred.linear[pred.linear <= 0] = -1
test.error.linear <- mean(pred.linear != testing$Y)
```

```
data.frame("train error"=train.error.linear, "test error"=test.error.linear)
```

```
##   train.error test.error
## 1 0.005759539 0.04120879
```

So training misclassification error and test misclassification error are 0.576% and 4.12%, respectively.

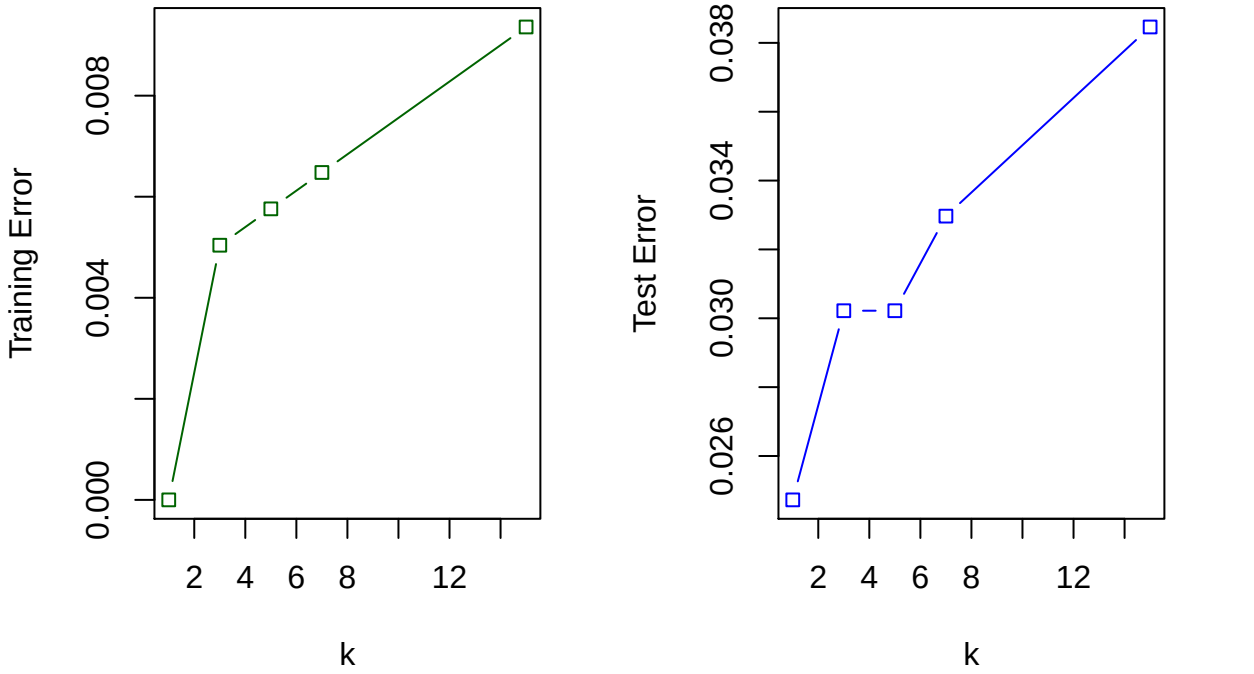
(b).

```
library(class)
k.vec <- c(1, 3, 5,7, 15)
train.error=rep(NA,5)
test.error=rep(NA,5)


# Calculate the training error and test error for each K
for(i in 1:5){
  knn.pred=knn(train = train[,1:256], test = testing[,1:256],
               cl = train[,257], k = k.vec[i])
  test.error[i]=mean(knn.pred!=testing[,257])

  knn.train=knn(train = train[,1:256], test = train[,1:256],
                cl = train[,257], k = k.vec[i])
  train.error[i]=mean(knn.train!=train[,257])
}
```

```r
# Plot the training errors and test errors
par(mfrow=c(1,2))
plot(k.vec, train.error,type="b", pch=22, col="darkgreen",xlab="k", ylab="Training Error")
plot(k.vec, test.error,type="b",pch=22, col="blue", xlab="k", ylab="Test Error")
```



In this case, it seems that test error increases with k. The 1-NN classifier delivers the lowest test error for this binary classification problem.