BST691 HW2

Ruijie Yin

Q1.

(a)Fit a LDA model and report training and testing errors

```
#generate training dataset
#Coding: 0-green, 1-red
library(MASS)
set.seed(2000)
miu1<-c(2,1)
miu2<-c(1,2)
sigma<-matrix(c(1,0,0,1),2,2,byrow = T)
green<-mvrnorm(100,miu1,sigma)
red<-mvrnorm(100,miu2,sigma)

#training dataset
label<-matrix(c(rep(0,100),rep(1,100)),200,1)
train_data<-rbind(green,red)
train_data<-cbind(train_data,label)
colnames(train_data)<-c("x1","x2","y")

#train a LDA model
#data must be a dataframe
train_data<-as.data.frame(train_data)
lda.obj<-lda(y~., train_data)

#traning error

lda.train<-predict(lda.obj,train_data[,-3])$class
lda.train<-as.numeric(lda.train)-1
lda.train.error<-(1-sum(lda.train==train_data[,3])/200)
lda.train.error
```

```
## [1] 0.22
```

```
#read in the testing set
#test_set<-read.table(file="C:/Users/yinru/Dropbox/UM Biostatistics/BST691
High Dimensional and #Complex Data/HW1/hw1q2testdataset.txt", header=F)
library(MASS)
set.seed(2014)
miu1<-c(2,1)
miu2<-c(1,2)
sigma<-matrix(c(1,0,0,1),2,2,byrow = T)
green2<-mvrnorm(500,miu1,sigma)
```

```r
red2<-mvrnorm(500,miu2,sigma)
label2<-matrix(c(rep(0,500),rep(1,500)),1000,1)
test_set<-rbind(green2,red2)
test_set<-cbind(test_set,label2)
colnames(test_set)<-c("x1","x2","y")

#testing error
#Note: <newdata> parameter needs to be dataframe
test_set<-as.data.frame(test_set)
lda.test<-predict(lda.obj,test_set[,-3])$class
lda.test<-as.numeric(lda.test)-1
lda.test.error<-(1-sum(lda.test==test_set[,3])/1000)
lda.test.error

## [1] 0.235
```

The training error is 0.22 and the testing error is 0.235

Q1.

(b)Fit a logistic regression model and report training and testing errors

```r
#train a logistic regression model
logistic.obj<-glm(y~.,family=binomial(link='logit'),data=train_data)
#training error
logistic.train<-predict(logistic.obj,x1=train_data[,-3],type='response')
logistic.train<-ifelse(logistic.train > 0.5,1,0)
logistic.train.error<-mean(logistic.train != train_data$y)
logistic.train.error

## [1] 0.215

#testing error
test_set<-as.data.frame(test_set)
logistic.test<-predict(logistic.obj,test_set[,-3],type='response')
logistic.test<-ifelse(logistic.test > 0.5,1,0)
logistic.test.error<-mean(logistic.test != test_set[,3])
logistic.test.error

## [1] 0.236
```

The training error is 0.215 and the testing error is 0.236.

Q1.

(c)

The training errors are similar for all four methods, however, the LDA and linear regression have the smallest testing errors, that is because when the underlying distribution is indeed normal, LDA is theoretically better than logistic regression, as the later one made no assumption on the distribution of the data; Besides, LDA and logistic regression does give similar results in binary classification scenarios.

Q2.Scenario 2

(a)

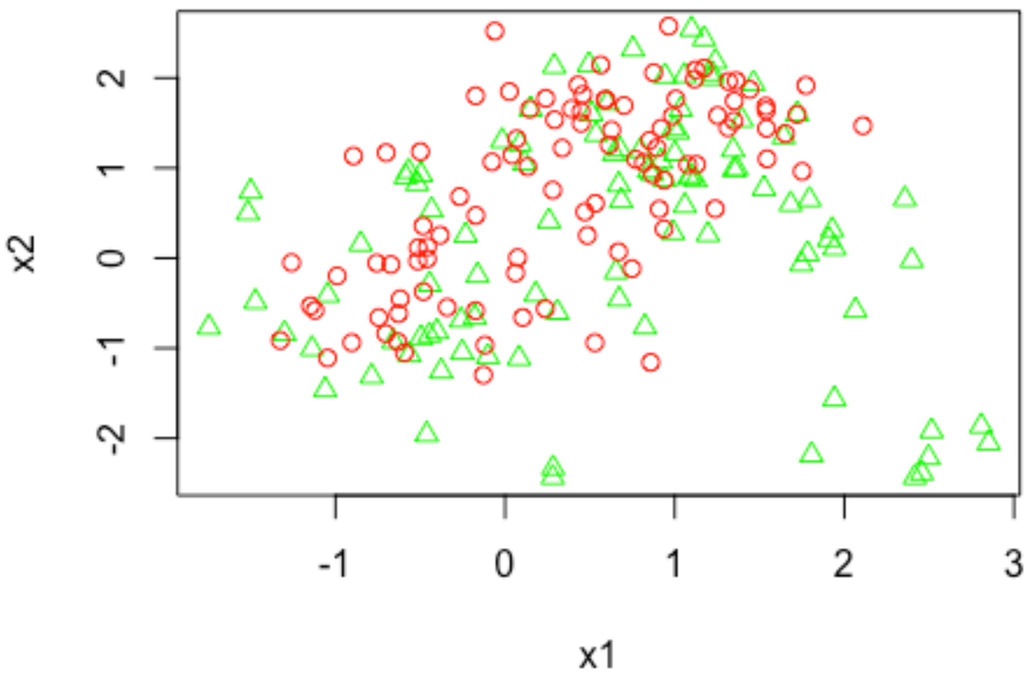
Generate a training set; Note: the seed is set to 2000.

```
library(MASS)
set.seed(2000)
miu1<-c(1,0)
sigma<-matrix(c(1,0,0,1),2,2,byrow = T)
miu_k<-mvrnorm(10,miu1,sigma)
miu2<-c(0,1)
v_k<-mvrnorm(10,miu2,sigma)

#green
green_train<-list()
for(i in 1:100) {
green_train[i]<-
list(mvrnorm(1,miu_k[sample(dim(miu_k)[1],1,prob=rep(1/10,10)),],sigma/5))
}
#convert green_train to a data frame
green_train <- data.frame(matrix(unlist(green_train), nrow=100, byrow=T))

#red
red_train<-list()
for(i in 1:100) {
red_train[i]<-
list(mvrnorm(1,v_k[sample(dim(v_k)[1],1,prob=rep(1/10,10)),],sigma/5))
}
#convert green_train to a data frame
red_train <- data.frame(matrix(unlist(red_train), nrow=100, byrow=T))

#training set
label<-matrix(c(rep(0,100),rep(1,100)),200,1)
training<-rbind(green_train,red_train)
training<-cbind(training,label)
colnames(training)<-c("x1","x2","y")
```

Q2.

(b)The red circles are in class ‘red’, while the green diamonds are in class ‘green’.

```
plot(green_train[,1],green_train[,2],col="green",pch=2,xlab = "x1",ylab =
"x2")
points(red_train[,1],red_train[,2],col="red")
```



Q2.

(c)Generate a test set

```
library(MASS)
set.seed(2019)
miu1<-c(1,0)
sigma<-matrix(c(1,0,0,1),2,2,byrow = T)
miu_k<-mvrnorm(10,miu1,sigma)
miu2<-c(0,1)
v_k<-mvrnorm(10,miu2,sigma)

#green
```

```r
green_test<-list()
for(i in 1:500) {
green_test[i]<-
list(mvrnorm(1,miu_k[sample(dim(miu_k)[1],1,prob=rep(1/10,10)),],sigma/5))
}
#convert green_test to a data frame
green_test <- data.frame(matrix(unlist(green_test), nrow=500, byrow=T))

#red
red_test<-list()
for(i in 1:500) {
red_test[i]<-
list(mvrnorm(1,v_k[sample(dim(v_k)[1],1,prob=rep(1/10,10)),],sigma/5))
}
#convert green_test to a data frame
red_test <- data.frame(matrix(unlist(red_test), nrow=500, byrow=T))

#testing set
label<-matrix(c(rep(0,500),rep(1,500)),1000,1)
testing<-rbind(green_test,red_test)
testing<-cbind(testing,label)

#save the testing set to local disk
write.table(testing,file="/Users/ruijieyin/Dropbox/UM Biostatistics/BST691
High Dimensional and Complex Data/HW2/test.txt")
```

Q3.

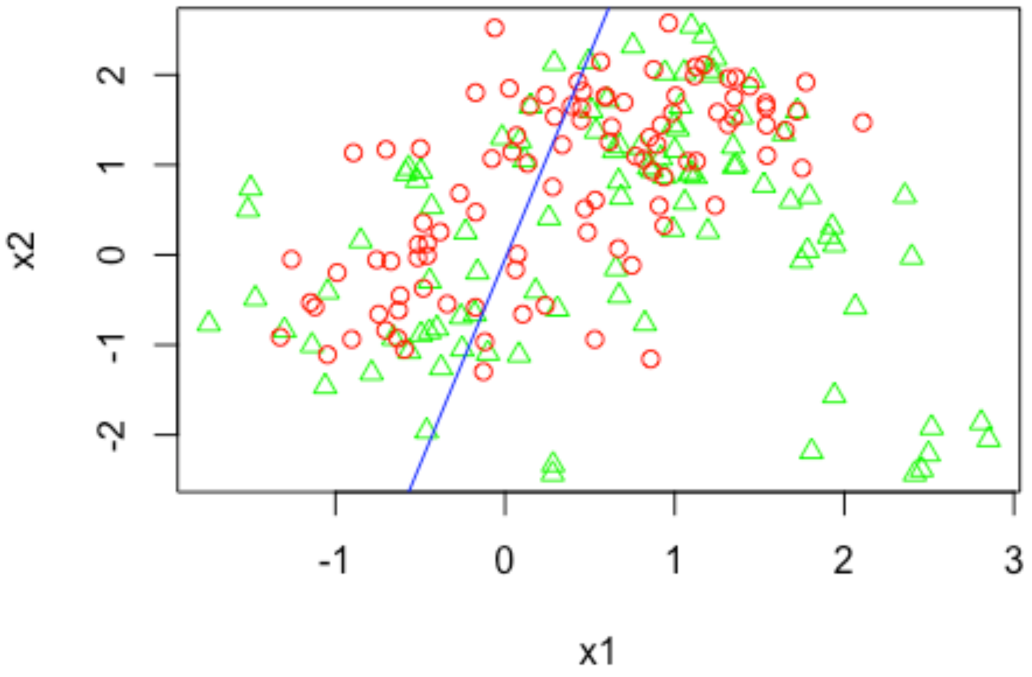
(a)Train the linear regression model on the training set

```r
linear_model<-lm(y~x1+x2,data = training)
coef<-coefficients(linear_model)
#print the fitted coefficients
coef

## (Intercept)          x1          x2
##   0.4950930  -0.1085670   0.1129672
```

Q3.

(b) Add the linear decision boundary line to the scatter plot.

```r
plot(green_train[,1],green_train[,2],col="green",pch=2,xlab = "x1",ylab =
"x2")
points(red_train[,1],red_train[,2],col="red")
#add the linear decision boundary line
abline(a=(0.5-coef[[1]])/coef[[2]],b=-coef[[1]]/coef[[2]],col="blue")
```



Q3

(c)Report training and testing error

```
#training error
linear.pred<-coef[[1]]+coef[[2]]*training[,1]+coef[[3]]*training[,2]-0.5
linear.pred<-as.matrix(linear.pred,200,1)
linear.result<-matrix(NA,200,1)
for(i in 1:200) {
  if(linear.pred[i,]>0) {
linear.result[i,]=1
}else {
  linear.result[i,]=0
}
}
linear.result<-as.numeric(linear.result)
linear.train.error<-as.numeric(training[,3]==linear.result)
linear.train.error<-1-sum(linear.train.error)/200
linear.train.error

## [1] 0.385
```

```
#testing error
testing<-read.table(file = "/Users/ruijieyin/Dropbox/UM Biostatistics/BST691
High Dimensional and Complex Data/HW2/test.txt", header=T)
testing<-as.data.frame(testing)
linear.pred.test<-coef[[1]]+coef[[2]]*testing[,1]+coef[[3]]*testing[,2]-0.5
linear.pred.test<-as.matrix(linear.pred.test,1000,1)
linear.result.test<-matrix(NA,1000,1)
for(i in 1:1000) {
  if(linear.pred.test[i,]>0) {
  linear.result.test[i,]=1
}else {
  linear.result.test[i,]=0
}
}
linear.result.test<-as.numeric(linear.result.test)
linear.test.error<-as.numeric(testing[,3]==linear.result.test)
linear.test.error<-1-sum(linear.test.error)/1000
linear.test.error

## [1] 0.219
```

The training error is 0.385 and the testing error is 0.219

Q4.

(a)Scenario 1

```
library(class)
#use train_data and test_set
#0:green 1:red
cl1<-factor(train_data[,3])
cl2<-factor(test_set[,3])
knn_scenario1_train_error<-matrix(NA,13,1)
knn_scenario1_test_error<-matrix(NA,13,1)
j=1
for (i in c(1,4,7,10,13,16,30,45,60,80,100,150,200)) {
    knn_scenario1_train<-knn(train_data[,-3],train_data[,-3],cl1,i)
    knn_scenario1_train_error[j,1]<-mean(knn_scenario1_train != cl1)
    knn_scenario1_test<-knn(train_data[,-3],test_set[,-3],cl1,i)
    knn_scenario1_test_error[j,1]<-mean(knn_scenario1_test != cl2)
    j=j+1
}
#generate a table to display the result
ks<-matrix(c(1,4,7,10,13,16,30,45,60,80,100,150,200),13,1)
scenario1_result<-
cbind(ks,knn_scenario1_train_error,knn_scenario1_test_error)
colnames(scenario1_result)<-c('k','training error','testing error')
rownames(scenario1_result)<-as.character(1:13)
```

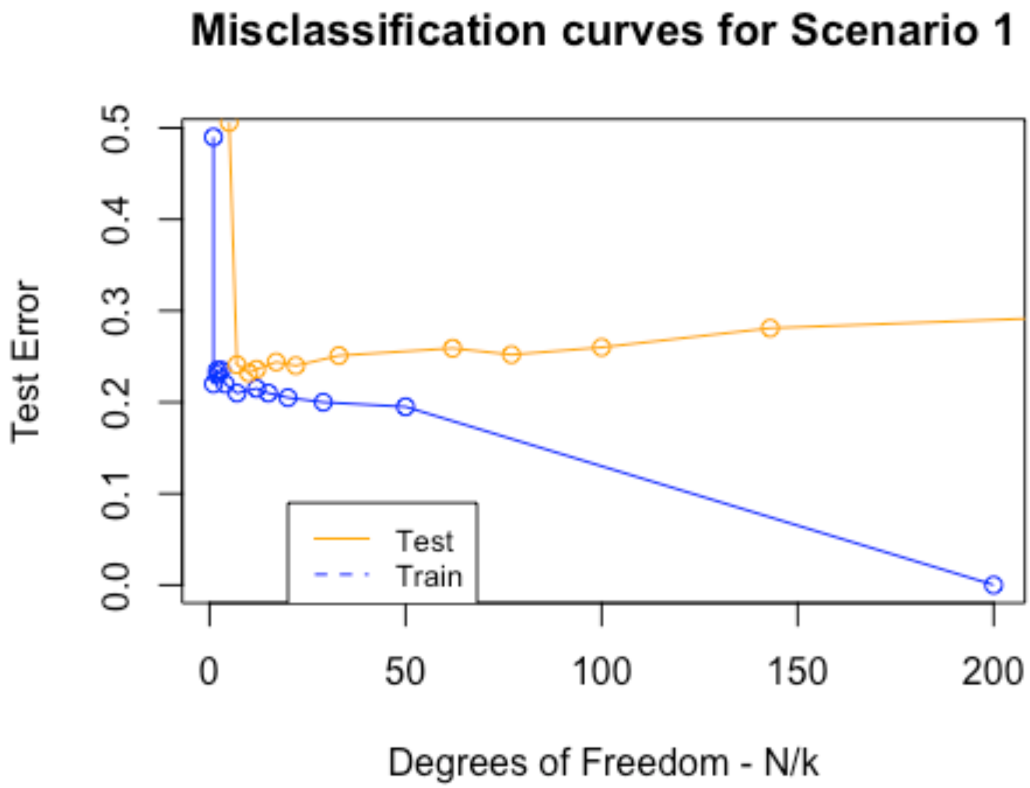
```
scenario1_result.table<-as.table(scenario1_result)
scenario1_result.table
```

```
##           k training error testing error
## 1    1.000          0.000         0.334
## 2    4.000          0.195         0.298
## 3    7.000          0.200         0.281
## 4   10.000          0.205         0.260
## 5   13.000          0.210         0.252
## 6   16.000          0.215         0.259
## 7   30.000          0.210         0.251
## 8   45.000          0.220         0.240
## 9   60.000          0.235         0.244
## 10  80.000          0.230         0.236
## 11 100.000          0.235         0.232
## 12 150.000          0.220         0.241
## 13 200.000          0.490         0.506
```

The training and testing error for each k is shown as the table above.

Plot the training error vs the degree of freedom n/k, and the testing error vs n/k, in one same figure

```
#degrees of freedom should be increasing
df1<-round(as.numeric(200/ks))
df1<-sort(df1,decreasing = F )
df2<-round(as.numeric(1000/ks))
df2<-sort(df2,decreasing = F )
plot(x=df1,y=rev(knn_scenario1_train_error),type = "o",col = "blue", xlab =
"Degrees of Freedom - N/k", ylab = "Test Error",main = "Misclassification
curves for Scenario 1")
lines(x=df2,y=rev(knn_scenario1_test_error),type = "o",col = "orange")
legend(20,0.09,legend=c("Test", "Train"),col=c("orange", "blue"), lty=1:2,
cex=0.8)
```



Q4.

(b) Repeat Q4.(a) for scenario 2

```
library(class)
#use dataset: training and testing
#0:green 1:red
cl3<-factor(training[,3])
cl4<-factor(testing[,3])
knn_scenario2_train_error<-matrix(NA,13,1)
knn_scenario2_test_error<-matrix(NA,13,1)
j=1
for (i in c(1,4,7,10,13,16,30,45,60,80,100,150,200)) {
    knn_scenario2_train<-knn(training[,-3],training[,-3],cl3,i)
    knn_scenario2_train_error[j,1]<-mean(knn_scenario2_train != cl3)
    knn_scenario2_test<-knn(training[,-3],testing[,-3],cl3,i)
    knn_scenario2_test_error[j,1]<-mean(knn_scenario2_test != cl4)
    j=j+1
}
#generate a table to display the result
ks<-matrix(c(1,4,7,10,13,16,30,45,60,80,100,150,200),13,1)
scenario2_result<-
```

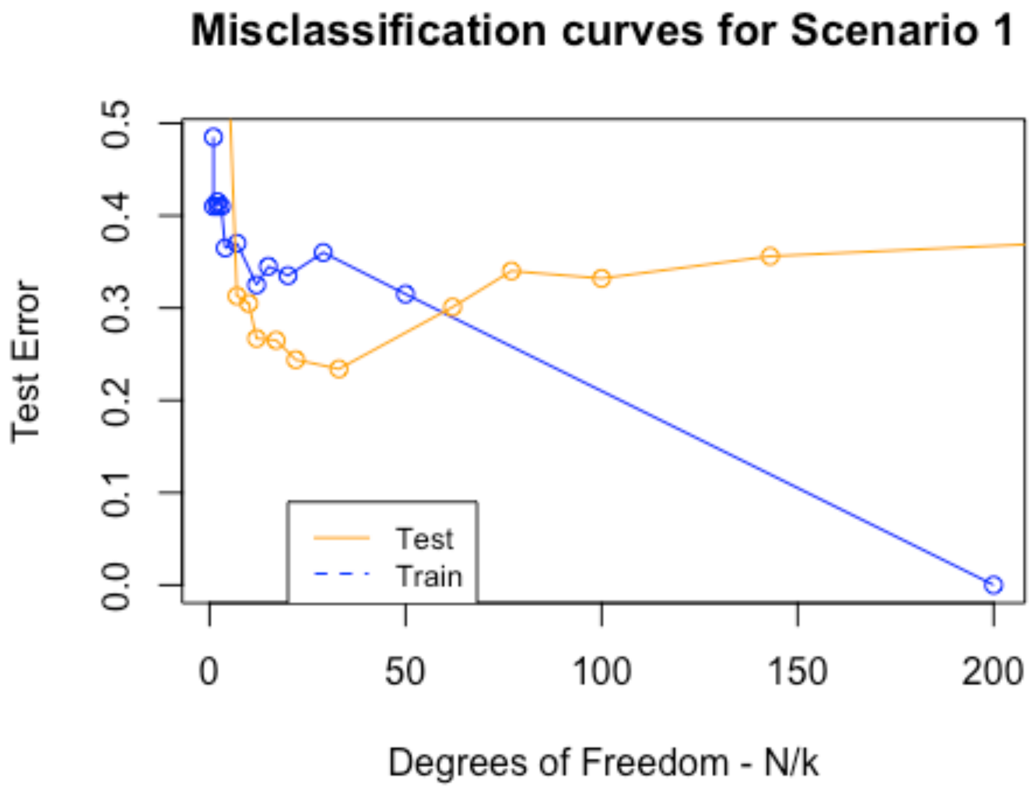
```
cbind(ks,knn_scenario2_train_error,knn_scenario2_test_error)
colnames(scenario2_result)<-c('k','training error','testing error')
rownames(scenario2_result)<-as.character(1:13)
scenario2_result.table<-as.table(scenario2_result)
scenario2_result.table
```

```
##           k training error testing error
## 1    1.000          0.000         0.350
## 2    4.000          0.315         0.377
## 3    7.000          0.360         0.356
## 4   10.000          0.335         0.332
## 5   13.000          0.345         0.340
## 6   16.000          0.325         0.301
## 7   30.000          0.370         0.234
## 8   45.000          0.365         0.244
## 9   60.000          0.410         0.265
## 10  80.000          0.415         0.267
## 11 100.000          0.410         0.305
## 12 150.000          0.410         0.313
## 13 200.000          0.485         0.538
```

The training and testing error for each k is shown as the table above.

Plot the training error vs the degree of freedom n/k, and the testing error vs n/k, in one same figure

```
#degrees of freedom should be increasing
df1<-round(as.numeric(200/ks))
df1<-sort(df1,decreasing = F )
df2<-round(as.numeric(1000/ks))
df2<-sort(df2,decreasing = F )
plot(x=df1,y=rev(knn_scenario2_train_error),type = "o",col = "blue", xlab =
"Degrees of Freedom - N/k", ylab = "Test Error",main = "Misclassification
curves for Scenario 1")
lines(x=df2,y=rev(knn_scenario2_test_error),type = "o",col = "orange")
legend(20,0.09,legend=c("Test", "Train"),col=c("orange", "blue"), lty=1:2,
cex=0.8)
```



Q4.

(c)

The training error decreases as the degrees of freedom increases, the training error is 0 when k=1 which indicates overfitting; The testing error, however, first decreases and then increases as degrees of freedom increases. The best k would be the k that corresponds to the smallest testing error in both scenarios, the k should be 100 for scenario 1 and 30 for scenario 2, respectively.