Mini Project #1

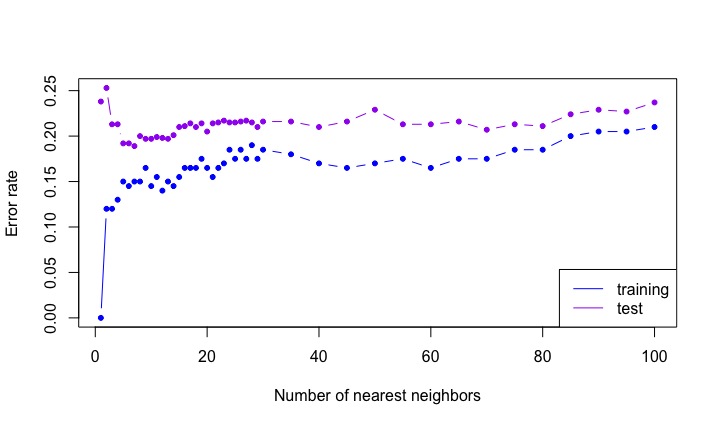
Name

Names of group members: Junmei Fan, Xi Cui

Contribution of each group member: 100% for Junmei Fan, 100% for Xi Cui

Section 1. Answers to the specific questions asked

(b)



We observed that the training error rate increases as the k increses, but the test error rate decreases first then once it reaches the minimum, it starts to increase. This is consistent with what we expect from the class: The number of neighbors K controls flexibility of the model, As K increases, flexibility decreases, implying that bias increases. We learned in the class that the training error rate decreases as the model flexibility increases but the test error rate shows a **U**-shape due to the bias-variance trade-off.

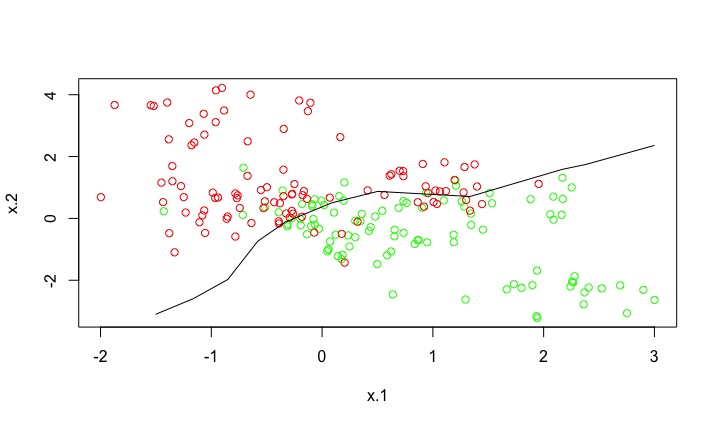
(c)

The optimal k is 7.

The training error rate associated with the optimal k is 0.15.

The testing error rate associated with the optimal k is 0.189.

(d)



Comment: With k=7, the decision boundary clearly separate the data into two classes. However around the boundary there are some misclassifying data, so the boundary is not very sensible. We should be very careful when classify the data around the boundary. It will be better if there is more information. If not, we should adjust the boundary according the actual needs.

Section 2. Code

#STAT 6340

#MINI Project1

#Group member: Junmei Fan & Xi Cui

setwd("/Users/junmeifan/Documents/Life/UTD\_School/2018spring/stat6340")

train=read.table('1-training\_data.csv',header=T,sep=',')

test=read.table('1-test\_data.csv', header=T, sep=',')

#look at the data

head(train)

head(test)

#look at the structure of the data

str(train)

#look at the summary of the data

summary(train)

#apply KNN

train.X <- train[,c(1:2)]

train.Y<- train$y

test.X<- test[, c(1:2)]

test.Y<-test$y

library(class)

# K = 1

set.seed(1)

mod.train <- knn(train.X, train.X, train.Y, k = 1)

table(mod.train, train.Y)

#train error rate

train.error=1-sum(mod.train==train.Y)/length(train.Y)

train.error

# Fit KNN for several values of K

ks <- c(seq(1, 30, by = 1), seq(35, 100, by = 5))

nks <- length(ks)

err.rate.train <- numeric(length = nks)

err.rate.test <- numeric(length = nks)

names(err.rate.train) <- names(err.rate.test) <- ks

for (i in seq(along = ks)) {

set.seed(1)

mod.train <- knn(train.X, train.X, train.Y, k = ks[i])

set.seed(1)

mod.test <- knn(train.X, test.X, train.Y, k = ks[i])

err.rate.train[i] <- 1 - sum(mod.train == train.Y)/length(train.Y)

err.rate.test[i] <- 1 - sum(mod.test == test.Y)/length(test.Y)

}

#Plot ks vs. err.rate.train/err.rate.test

plot(ks, err.rate.train, xlab = "Number of nearest neighbors", ylab = "Error rate",

type = "b", ylim = range(c(err.rate.train, err.rate.test)), col = "blue", pch = 20)

lines(ks, err.rate.test, type="b", col="purple", pch = 20)

legend("bottomright", lty = 1, col = c("blue", "purple"), legend = c("training", "test"))

#find the optimal k, the training and test error rate associated with it

result <- data.frame(ks, err.rate.train, err.rate.test)

result[err.rate.test == min(result$err.rate.test), ]

# Decision boundary for optimal K

n.grid <- 7

x1.grid <- seq(f = min(train.X[, 1]), t = max(train.X[, 1]), l = n.grid)

x2.grid <- seq(f = min(train.X[, 2]), t = max(train.X[, 2]), l = n.grid)

grid <- expand.grid(x1.grid, x2.grid)

k.opt <- 7

set.seed(1)

mod.opt <- knn(train.X, grid, train.Y, k = k.opt, prob = T)

prob <- attr(mod.opt, "prob") # prob is voting fraction for winning class

prob <- ifelse(mod.opt == "yes", prob, 1 - prob) # now it is voting fraction for y == "yes"

prob <- matrix(prob, n.grid, n.grid)

plot(train.X, col = ifelse(train.Y == "yes", "green", "red"))

contour(x1.grid, x2.grid, prob, levels = 0.5, labels = "", xlab = "", ylab = "",

main = "", add = T)