STAT 6340 (Statistical and Machine Learning), Spring 2019

Mini Project 1 (Solution)

February 22, 2019

1. Here we provide the answers to the questions asked. The relevant `R` code is presented in the `R` code section.

   (a) The `R` code section shows how KNN was fit for $K = 1, 2, \ldots, 30, 35, \ldots, 100$ and how the test and training error rates were calculated.

   (b) Figure 1 shows the plots of training and test error rates against the number of nearest neighbors K. These curves exhibit some fluctuations and are not as smooth looking as we saw in the class. We observe that the training error rate more or less increases as the model flexibility decreases, i.e., as K increases. This is consistent with what we learned in class. However, the test error rate does not clearly exhibit the expected U-shape. If the range of K is increased, the desired behavior of the test error rate will be observed.

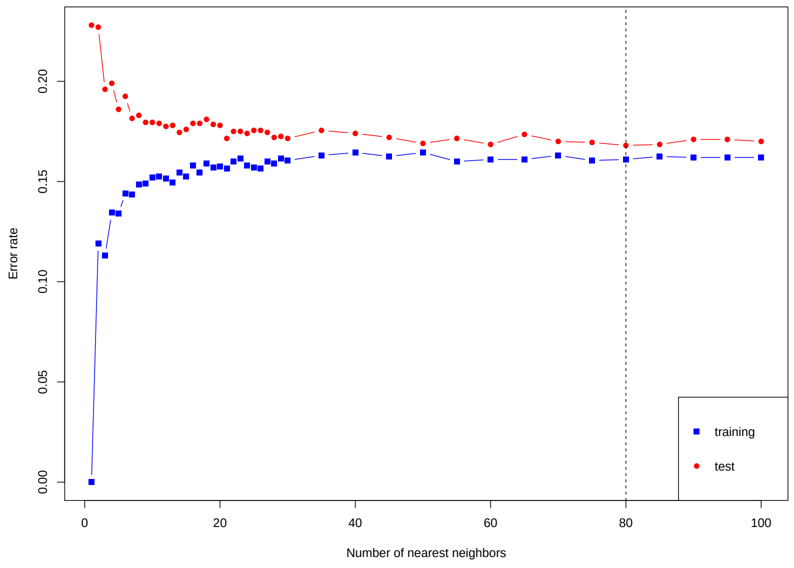

Figure 1: Plots of training and test error rates against the number of nearest neighbors K. The test error rate is minimized at $K = 80$.

   (c) From the plot of test error rate in Figure 1, it can be observed that the optimal value of K is 80 – where the test error rate attains its minimum. However, notice that the test error rate more or less stabilizes after $K = 35$. For $K = 80$, the associated training and test error rates are 0.161 and 0.168 respectively (the numbers may slightly vary due to using a different seed value).

(d) Figure 2 shows the desired plot of the data together with the decision boundary for $K = 80$, which is the optimal choice. The decision boundary (being close to be linear) is able to separate the majority of `yes` and `no` classes despite the presence of several misclassified points. Therefore, it appears sensible.

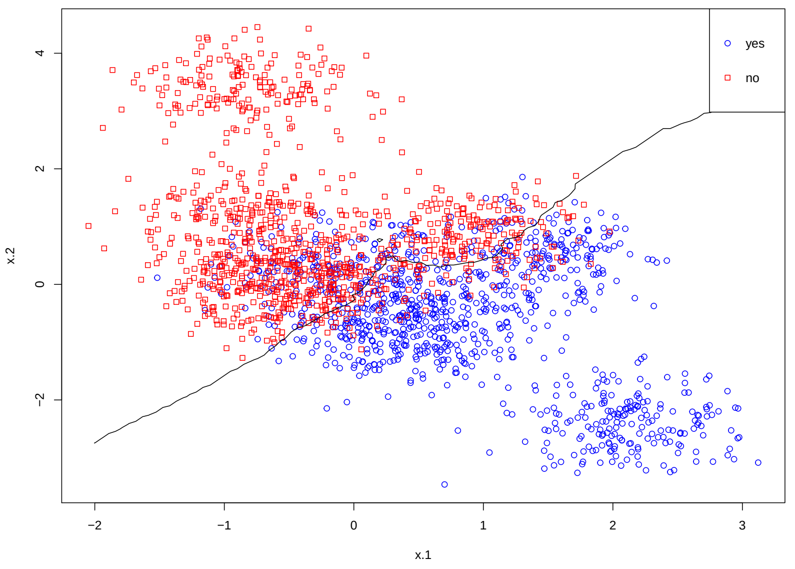

Figure 2: A plot of the training data together with the decision boundary for the optimal K (i.e., $K = 80$).

```
# Load training data
train.Data = read.csv('1-training_data.csv',header = TRUE)
head(train.Data)
attach(train.Data)
# Matrix of training set cases
train.X = cbind(x.1,x.2)
# Factor of true classes for training set
train.Y = y
detach(train.Data)

# Load test data
test.Data=read.csv('1-test_data.csv',header = TRUE)
head(test.Data)
attach(test.Data)
# Matrix of test set cases
test.X = cbind(x.1,x.2)
# Factor of true classes for test set
test.Y = y
```

```
detach(test.Data)

# Apply KNN
library(class)

# (a) Fit KNN with K = 1,2,...,30,35,...,100.

# Form sequence of K's
ks=c(seq(1, 30, by = 1), seq(35, 100, by = 5))
# Define variables to record
err.rate.train=err.rate.test=c()

for (i in seq(along = ks)) {
set.seed(1)
# Fit KNN for K_i and evaluate its performance on training set
mod.train <- knn(train=train.X, test=train.X, cl=train.Y, k = ks[i])
set.seed(1)
# Fit KNN for K_i and evaluate its performance on test set
mod.test <- knn(train.X, test.X, train.Y, k = ks[i])
# Record error rate for training set
err.rate.train[i] <- 1 - mean(mod.train == train.Y)
# Record error rate for test set
err.rate.test[i] <- 1 - mean(mod.test == test.Y)
}

# (b) Plot training and test error rates against K. Explain what you observe.
# Is it consistent with what you expect from the class?

plot(ks, err.rate.train, xlab = "Number of nearest neighbors", ylab = "Error rate",
type = "b", ylim = range(c(err.rate.train, err.rate.test)), pch = 15,col='blue')
lines(ks, err.rate.test, type="b", pch = 16,col='red')
legend("bottomright", pch = c(15,16), legend = c("training", "test"),col=c("blue","red"))
abline(v=ks[which.min(err.rate.test)],lty = "dashed")

# (c) What is the optimal value of K? What are the training and
# test error rates associated with the optimal K?

# print training and test error rates for the optimal K
result=data.frame(ks, err.rate.train, err.rate.test)
result[err.rate.test == min(result$err.rate.test),]

# (d) Make a plot of the training data that also shows the decision boundary for the
# optimal K. Comment on what you observe. Does the decision boundary seem sensible?

# Form grid
n.grid <- 100
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)

# Record optimal value of K (if more than two, we take the smallest)
k.opt <- 80

set.seed(1)
# Fit KNN for optimal K and evaluate its performance on points belonging to grid
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)
prob <- matrix(prob, n.grid, n.grid)

plot(train.X, pch = ifelse(train.Y == "yes", 21, 22),
col=ifelse(train.Y == "yes", "blue","red"))
contour(x1.grid, x2.grid, prob, levels = 0.5, labels = "",
xlab = "", ylab = "",main = "", add = T)
legend("topright", pch = c(21,22), legend = c("yes", "no"), col=c('blue','red'))
```