Project 01

Indrajith Wasala Mudiyanselage

Question 01

a) KNN model was fitted for K=1,6,11,. . . ,196. The error rates for both training and test data was recorded.

b) By looking at the graph, as K increases the training error rate increases. In other words, as flexibility (1/K) increases training error rate decreases. On the other hand, the test error rate rapidly decreases and then increases slowly (Has the expected U shape). Therefore this is consistent with what we expect from the class.

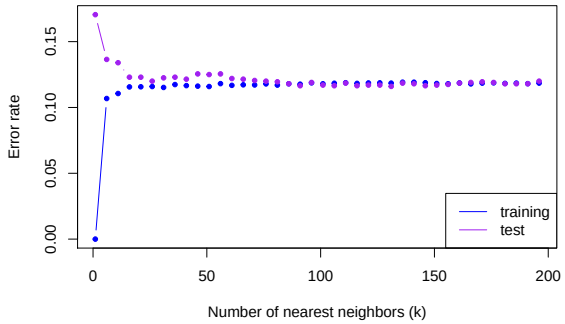

Figure 1: Training and test error rates against K

c) Optimal K value is 131 (When test error rate is minimum).

K value	Train error rate	Test error rate
131	0.1185	0.116

Table 1: Corresponding values to minimum test error rate.

d) By looking at the graph we can see there are two clusters. In each cluster, for lower X2 values, we get “yes” as the response (Y), and for higher X2 values we get ”no" as the response (Y). The decision bounder seems to do a good job, as it separates most of the points in the test data correctly. Therefore the decision boundary seems sensible.

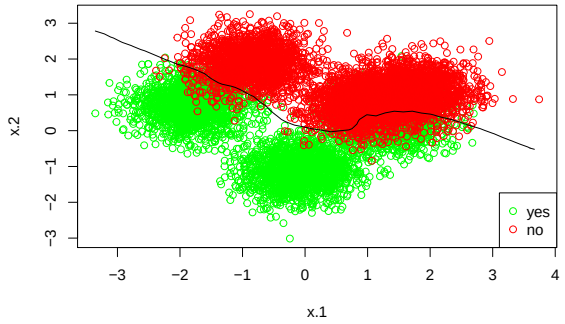

Figure 2: Plot of the training data with decision boundary

Question 02

a) KNN model was fitted for K={50, 100, 200, 300, 400}. The error rates for test data was recorded.There is no significant difference in the test error rates for different K values. Note that even though we have small differences in test error rates, we still get the U-shape that we expected from the class.

K value	50	100	200	300	400
Test error rate	0.67	0.66	0.65	0.66	0.67

Table 2: Test error rates according to K.

b) Select K=200 as the best K value even though it has a very small error rate difference with other values. This is a fair discretion because the flexibility of the model is also kind of in the middle range for the selected K values.

	Predicted class									
True class	0	1	2	3	4	5	6	7	8	9
0	2	0	1	0	0	0	2	0	1	0
1	0	1	2	0	0	0	2	0	2	1
2	0	0	3	0	1	0	1	0	0	0
3	0	1	2	2	5	1	0	0	0	0
4	1	0	4	0	7	0	0	0	0	0
5	0	0	3	0	1	1	2	0	1	1
6	0	0	1	0	5	0	6	0	0	0
7	0	0	0	1	3	1	0	1	1	0
8	0	0	0	0	2	1	1	0	14	1
9	0	0	2	0	4	0	0	0	3	2

Table 3: Confusion matrix

By looking at the confusion matrix we can clearly see the number of misclassifications is very high (~65%). Therefore KNN is not suitable as an image classification method.

c) In an image datasets, the number of image features (variables) is very high (In this example we have $32 \times 32 \times 3 = 3072$) . Therefore, the dimension of each data point is also very high. Since KNN is using distance measures (like euclidean distance), when the number of dimension increases it is harder to identify two points are close to each other on every dimension. Therefore the accuracy of the classification is getting lower. On the other hand, KNN is very time-consuming if we have more observations in image data. Therefore KNN is an inefficient method in image classification.

Question 03

a)

$$\begin{aligned}
MSE\{\hat{f}(x_0)\} &= E\{\hat{f}(x_0) - f(x_0)\}^2 = E\{(\hat{f}(x_0) - E[\hat{f}(x_0)]) + (E[\hat{f}(x_0)] - f(x_0))\}^2 \\
&= E\{(\hat{f}(x_0) - E[\hat{f}(x_0)])^2 + (E[\hat{f}(x_0)] - f(x_0))^2 + 2(\hat{f}(x_0) - E[\hat{f}(x_0)])(E[\hat{f}(x_0)] - f(x_0))\} \\
&= E\{\hat{f}(x_0) - E[\hat{f}(x_0)]\}^2 + \{E[\hat{f}(x_0)] - f(x_0)\}^2 + 2(E[\hat{f}(x_0)] - E[\hat{f}(x_0)])(E[\hat{f}(x_0)] - f(x_0)) \\
&= Var\{\hat{x_0}\} + (Bias\{\hat{f}(x_0)\})^2 + 0 = (Bias\{\hat{f}(x_0)\})^2 + Var\{\hat{x_0}\}
\end{aligned}$$

b)

$$\begin{aligned}
E(\hat{Y}_0 - Y_0)^2 &= E\{\hat{f}(x_0) - (f(x_0) + \epsilon_0)\}^2 = E\{(\hat{f}(x_0) - f(x_0)) - \epsilon_0\}^2 \\
&= E\{(\hat{f}(x_0) - f(x_0))^2 + \epsilon_0^2 - 2(\hat{f}(x_0) - f(x_0))\epsilon_0\} = E\{\hat{f}(x_0) - f(x_0)\}^2 + E\{\epsilon_0^2\} - 0 \\
&= MSE\{\hat{f}(x_0)\} + Var\{\epsilon_0\} + \{E[\epsilon_0]\}^2 = (Bias\{\hat{f}(x_0)\})^2 + Var\{\hat{x_0}\} + Var\{\epsilon_0\} \\
&= (Bias\{\hat{f}(x_0)\})^2 + Var\{\hat{x_0}\} + \sigma^2
\end{aligned}$$

R Codes

```
knitr::opts_chunk$set(echo = TRUE)


## ----include=FALSE-----------------------------------------------------------------------------------------------------
setwd("D:/UTD/Academic/5-Spring 2021/STAT 6340/Projects/Project 01")

### Question 01

## ----include=FALSE-----------------------------------------------------------------------------------------------------
training<-read.csv("1-training_data.csv",header = T)  # Import training data to R
str(training)
test<-read.csv("1-test_data.csv",header = T)  # Import test data to R
str(test)

train.X<-training[,1:2]  # trining X data
train.Y<-training[,3]  # trining Y data

test.X<-test[,1:2]  # test X data
test.Y<-test[,3]  # test Y data

### a)

## ----include=FALSE-----------------------------------------------------------------------------------------------------
#Fit KNN for several values of K (K=1,2,...,196)

ks <- c(seq(1, 200, 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


## ---- cache=TRUE, echo=FALSE------------------------------------------------------------------------------------------
library(class)

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] <- mean(mod.train != train.Y)
  err.rate.test[i] <- mean(mod.test != test.Y)
}

### b)

## ----echo=FALSE, out.width = "40%", fig.align="center",fig.cap="Training and test error rates against K"----------
plot(ks, err.rate.train, xlab = "Number of nearest neighbors (k)", ylab = "Error rate", type = "b", ylim = range
lines(ks, err.rate.test, type="b", col="purple", pch = 20)
legend("bottomright", lty = 1, col = c("blue", "purple"), legend = c("training", "test"))

### c)

## ----include=FALSE-----------------------------------------------------------------------------------------------------
library(xtable)
result <- data.frame(ks, err.rate.train, err.rate.test)
result[err.rate.test == min(result$err.rate.test),]

### d)

## ----echo=FALSE, out.width = "40%", fig.align="center",fig.cap="Plot of the training data with decision bounda
```

```r
library(class)
#Decision boundary for optimal K

n.grid <- 50
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 <- 131
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 Direction == "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)
legend("bottomright", pch = 1, col = c("green", "red"), legend = c("yes", "no"))

### Question 02

## ----include=FALSE-----------------------------------------------------------------------------------------------------------
library(keras)
cifar <- dataset_cifar10()
str(cifar)
x.train <- cifar$train$x
y.train <- cifar$train$y
x.test <- cifar$test$x
y.test <- cifar$test$y
# reshape the images as vectors (column-wise)
# (aka flatten or convert into wide format)
# (for row-wise reshaping, see ?array_reshape)
dim(x.train) <- c(nrow(x.train), 32*32*3) # 50000 x 3072
dim(x.test) <- c(nrow(x.test), 32*32*3) # 50000 x 3072
# rescale the x to lie between 0 and 1
x.train <- x.train/255
x.test <- x.test/255
# categorize the response
y.train <- as.factor(y.train)
y.test <- as.factor(y.test)
# randomly sample 1/100 of test data to reduce computing time
set.seed(2021)
id.test <- sample(1:10000, 100)
x.test <- x.test[id.test,]
y.test <- y.test[id.test]

### a)

## ----include=FALSE-----------------------------------------------------------------------------------------------------------
#Fit KNN for several values of K (K=50, 100, 200, 300, 400)

ks2<-c(50, 100, 200, 300, 400)
nks2 <- length(ks2)
err.rate.test2 <- numeric(length = nks2)
names(err.rate.test2) <- ks2


## ---- cache=TRUE, echo=FALSE------------------------------------------------------------------------------------------------
library(class)


for (i in seq(along = ks2)) {
```

```r
  set.seed(1)
  mod.test2 <- knn(x.train, x.test, y.train, k = ks2[i])
  err.rate.test2[i] <- mean(mod.test2 != y.test)
}

### b)

## ----cache=TRUE, echo=FALSE----------------------------------------------------------------------------------------
set.seed(1)
mod.test.K200 <- knn(x.train, x.test, y.train, k = 200)
err.rate.test.k200 <- mean(mod.test.K200 != y.test)


## ----include=FALSE-------------------------------------------------------------------------------------------------
con.mat<-table(y.test,mod.test.K200)
sum(diag(con.mat))



## ---- echo=FALSE, results="asis"-----------------------------------------------------------------------------------
library(xtable)
options(xtable.comment=FALSE)
row<-list()
row$pos<-list(0,0)
row$command<-c("& \\multicolumn{10}{c}{Perdicted class}\\\\\n","True class & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 &
print(xtable(con.mat, caption = "Confusion matrix"),add.to.row = row,include.colnames = FALSE,table.placement =
```