STAT 6340 Statistical Machine Learning

Mini Project 1

Author:
Suarez Rodes, Randy

Supervisor:
Choudhary, Pankaj Ph.D.

February 9, 2021

1 Answers

Question 1.a
We have fitted KNN for both training and testing set for values of k ranging between 1, 5, ..., 196 and additional values of 200, 225, 250, ..., 400.

Question 1.b
We present the training and testing error rates against the number of nearest neighbors on Figure 1. We can appreciate that for small values of k we have a low error rates on the training set and high error rates for the testing set. As the value of K increases the error of the training set increases while the testing set start to decrease. Both training and testing sets stabilizes and then the testing set increases again. This is consistent with class discussion since the KNN algorithm flexibility grows proportionally inverse to the value of K. For small values of K this algorithm tends to overfit due to high variance. As K increases, flexibility reduces, implying that bias rises and variance falls due to the bias-variance trade off, therefore the test error rate graph starts to show an U shape.

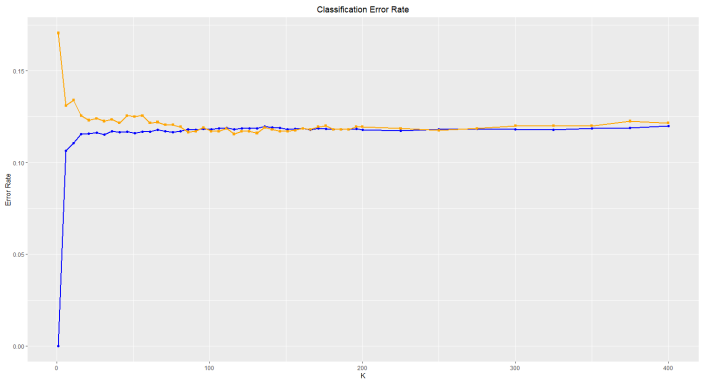

Figure 1: Classification Error Rates

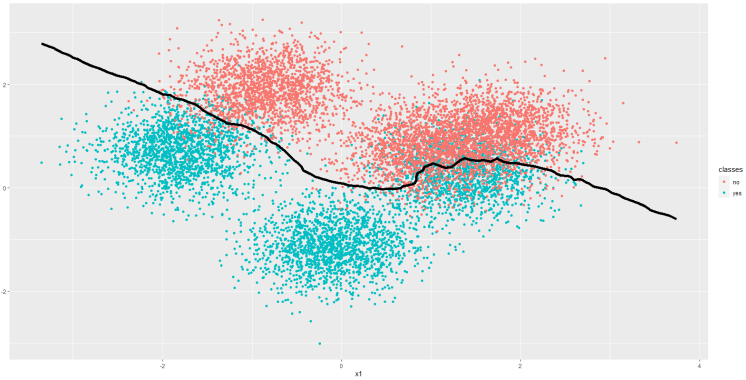

Figure 2: Decision Boundary K = 116

Question 1.c
The optimal K for our experiment is 116, dealing a training error rate of 0.1181 and a testing error rate of 0.1155.

Question 1.d
We present the training data set along with the decision boundary obtained from the optimal value K(116) on Figure 2. We appreciate how the decision boundary separates the training set into two well defined groups and the majority of the observations are classified correctly. There is some misclassification close to the decision boundary, this is expected, since the error rate for this value of K is around the 12%. So we can affirm that the decision boundary tends to predict the correct class with an 88% accuracy. Therefore we conclude that the decision boundary it is not as much sensible.

Question 2.a
We have fitted KNN for the testing set for values of K = 50, 100, 200, 300 and 400. The error rates are presented on Table 1. Observe the high error rates for the different values of K. Concluding that the KNN classifier did not perform well on the testing set.

Question 2.b
We present the confusion matrix for the best value of K(50) on Table 2. A quick look at the confusion matrix immediately shows that the misclassification is high. In particular we can quickly find the accuracy yielding 36% which is as

bad as random guess. We can notice that many observations have been wrongly classified as class 2 (birds), only one observation from class 7 (horses) has been correctly classified and also only one for class 1 (automobile).

Table 1: Test Error Rate

kval	Error
50	0.64
100	0.67
200	0.67
300	0.66
400	0.67

Table 2: Confusion Matrix

		Actual									
		0	1	2	3	4	5	6	7	8	9
Predicted	0	2	1	0	0	0	0	1	0	1	0
	1	0	1	0	0	0	0	0	0	1	0
	2	2	1	3	4	6	5	1	2	2	3
	3	0	0	0	1	0	0	0	0	0	0
	4	0	0	1	4	5	0	2	2	0	3
	5	0	0	0	1	0	1	0	1	0	0
	6	1	2	1	1	0	1	8	0	2	0
	7	0	0	0	0	0	0	0	1	0	0
	8	1	2	0	0	1	2	0	1	12	3
	9	0	1	0	0	0	0	0	0	1	2

Question 2.c
It is not a good idea to use KNN for image classification since the KNN algorithm relies on a distance metric to find similarities. This is a problem for high-dimensional objects like images, since distances over high dimensional spaces can be very counter intuitive. Also, depending on the distance, some transformation might be require leading to possible loss of information. Another problem is that KNN considers all features equally important for computing the similarities of observations. This can lead to misclassification in many situations where the algorithm determines two objects are close using irrelevant features like color or size in many cases.

Question 3.a.b
By definition: $MSE\{\hat{f}(x_0)\} = E[\hat{f}(x_0) - f(x_0)]^2$,
$Bias\{\hat{f}(x_0)\} = E[\hat{f}(x_0)] - f(x_0)$ and $var\{\hat{f}(x_0)\} = E(\hat{f}(x_0)^2 - E[f(x_0)])^2$

Then: $(Bias\{\hat{f}(x_0)\})^2 = E[\hat{f}(x_0)]^2 - 2E[\hat{f}(x_0)]f(x_0) + f(x_0)^2$

$var\{\hat{f}(x_0)\} = E([\hat{f}(x_0)^2 - E[\hat{f}(x_0)][\hat{f}(x_0)^2 - E[\hat{f}(x_0)]) = E[\hat{f}(x_0)^2] - 2E[\hat{f}(x_0)]E[\hat{f}(x_0)] + E[\hat{f}(x_0)]^2 = E[\hat{f}(x_0)^2] - E[\hat{f}(x_0)]^2$

Combining the Bias and Variance terms we obtain: $E[\hat{f}(x_0)]^2 - 2E[\hat{f}(x_0)]f(x_0) + f(x_0)^2 + E[\hat{f}(x_0)^2] - E[\hat{f}(x_0)]^2 = E[\hat{f}(x_0)^2] - 2E[\hat{f}(x_0)]f(x_0) + f(x_0)^2$

Also note that: $MSE\{\hat{f}(x_0)\} = E[\hat{f}(x_0) - f(x_0)]^2 = E[\hat{f}(x_0)^2] - 2E[\hat{f}(x_0)f(x_0)] + E[f(x_0)^2] = E[\hat{f}(x_0)^2] - 2E[\hat{f}(x_0)]f(x_0) + f(x_0)^2$
Since $f(x_0)$ is not a random variable and its expectation is equal to itself.

Therefore from both expressions we have that $MSE\{\hat{f}(x_0)\} = (Bias\{\hat{f}(x_0)\})^2 + var\{\hat{f}(x_0)\}$

Now by definition: $Y_0 = f(x_0) + \epsilon_0$, $\hat{Y}_0 = \hat{f}(x_0)$
and $\sigma^2 = E[\epsilon_0^2 - E(\epsilon_0)]^2 = E(\epsilon_0^2) - [E(\epsilon_0)]^2 = E(\epsilon_0^2)$, since $E(\epsilon_0) = 0$. Then:

$E(\hat{Y}_0 - Y_0)^2 = E[\hat{f}(x_0) - f(x_0) - \epsilon_0]^2 = E[\hat{f}(x_0) - f(x_0)]^2 - 2E\{[\hat{f}(x_0) - f(x_0)]\epsilon_0\} + E(\epsilon_0^2) =$

$MSE\{\hat{f}(x_0)\} - 2E[(\hat{f}(x_0) - f(x_0)]E[\epsilon_0] + \sigma^2 = MSE\{\hat{f}(x_0)\} + \sigma^2$ QED.
Where in the last two steps we have used the independence of ϵ_0 and the fact that $E(\epsilon_0) = 0$

2 Code

```
#Student Name: Randy Suarez Rodes
#Course: STAT 6340
#Mini Project 1
#February 9, 2021
#R version 4.0.3


#Packages
library(ggplot2) #Used for graphics an visual representations
library(class) #Used for KNN models

rdseed=8467 #Seed to replicate results in case of a tie on KNN

#Helper function to calculate the classification error rate
classification_error_rate=function(ypred,ytrue)
{
  mean(ypred!=ytrue)
}


#Setting graphic options.
error_colors=c("blue","orange")
graph_legend=c("Training Set","Testing Set")


##########################


#Experiment 1

#Values of K for experiment 1
kvals=c(seq(1,200,5),seq(200,400,25))


#Reading training and testing data sets
trn=read.csv("1-training_data.csv", stringsAsFactors = TRUE)
tst=read.csv("1-test_data.csv", stringsAsFactors = TRUE)

#Exploring the data set
str(trn)
print(summary(trn))

str(tst)
print(summary(tst))
#We can appreciate that the class labels are equally distributed
#in both training and testing sets
#Also normalization is not required since the attributes are in similar scale

#Saving training and testing labels
trn_y=trn$y
tst_y=tst$y

#Dropping the classes to use only the predictors on the knn function.
trn$y=NULL
tst$y=NULL

#Dataframe to track the error rates
Error_df=data.frame(kval=kvals)

#Question 1.a
```

```r
#Fitting KNN for different values of K (training set)
Error_df$trn_Error = sapply(kvals, function(i){
  set.seed(rdseed)
  trn_pred = knn(trn,trn,cl=trn_y,k=i)

  (classification_error_rate(trn_pred,trn_y))
})

#Fitting KNN for different values of K (testing set)
Error_df$tst_Error = sapply(kvals, function(i){
  set.seed(rdseed)
  tst_pred = knn(trn,tst,cl=trn_y,k=i)

  (classification_error_rate(tst_pred,tst_y))
})


#Question 1.b

#Generating plot of training and testing error rates against k.

g=ggplot(data=Error_df, aes(x=kval, y=trn_Error))+
  geom_line(aes(y=trn_Error, color=graph_legend[1]), size=1)+
  geom_point(color=error_colors[1], shape=19)+
  geom_line(aes(y=tst_Error, color=graph_legend[2]), size=1)+
  geom_point(x=Error_df$kval, y=Error_df$tst_Error,
             color=error_colors[2], shape=15)+
  scale_color_manual("Legend",values=c("Training Set"=error_colors[1],
                                       "Testing Set"=error_colors[2]))+
  labs(title="Classification Error Rate", x="K", y="Error Rate")+
  theme(plot.title=element_text(hjust=0.5))

print(g)

#Question 1.c

#Finding the index of the optimal K, this is the
#index of the least test error rate
ind_optimalK=which.min(Error_df$tst_Error)

print(Error_df[ind_optimalK,]) #The row of the optimal k contains
                        #the associated errors for training and testing.

optimalK=Error_df$kval[ind_optimalK]


#Question 1.d

#Creating grid
x1=seq(min(trn[,1]),max(trn[,1]),length.out=100)
x2=seq(min(trn[,2]),max(trn[,2]),length.out=100)
grid <- expand.grid(x=x1, y=x2)

#Classifying the grid
bestK=knn(trn,grid,cl=trn_y,k=optimalK,prob = TRUE ) #Fitting with optimal K
prob = attr(bestK, "prob")
prob = ifelse(bestK=="yes", prob, 1-prob)

#Data Frame to generate the surface for the contour
df_contour=data.frame(x=grid[,1],y=grid[,2],z=prob)

#Data Frame to plot the training set
df_trn=data.frame(x1=trn[,1],x2=trn[,2],classes=trn_y)
```

```r
#Plotting the training set and decision boundary
g2=ggplot()+geom_point(aes(x=x1,y=x2,color=classes),data=df_trn)+
  geom_contour(aes(x=x,y=y,z=z),
               data=df_contour,size=2,colour="black",breaks = 0.5)
print(g2)

##########################

#Experiment 2

##Preprocessing
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]

set.seed(1234)
id.train <- sample(1:50000, 5000)

x.train <- x.train[id.train,]
y.train <- y.train[id.train]

#Values of K for experiment 2
kvals2=c(50, 100, 200, 300, 400)

#Dataframe to track the test error rates
cifar_Error=data.frame(kval=kvals2)

##Experiment 2 questions

#Question 2.a

#Fitting KNN for different values of K
cifar_Error$Error = sapply(kvals2, function(i){
  set.seed(rdseed) #setting seeds for unties
  tst_pred = knn(x.train,x.test,cl=y.train,k=i)
```

```
  (classification_error_rate(tst_pred,y.test))

})

#Displaying Test Error Rates
print(cifar_Error)

#Question 2.b

ind_optimalK=which.min(cifar_Error$Error) #Finding index of optimal K
optimalK2=cifar_Error$kval[ind_optimalK]


#Fitting KNN with the optimal K
set.seed(rdseed)
cifar_pred=knn(x.train,x.test,cl=y.train,k=optimalK2)

#Displaying confusion matrix
print(table(cifar_pred ,y.test,dnn=c("predicted","actual")))
```