MATH 6350: Statistical Learning and Data Mining
Homework 3 Part 2

Contributions of Co-Authors

Stephanie Dinh (sdinh@central.uh.edu): Co-wrote part 1.
Ying-Yu Huang (yingyu010365@gmail.com): Co-wrote part 1.
Patricia Sieng (patricia.sieng@yahoo.com): Wrote the code and interpretation for Part 2.

Part 2: Data Analysis

Question 1

a = The smallest integer j such that Rj > 35%

a = 5

$R_5 = 35.02\%$

b = The smallest integer j such that Rj > 60%

b = 16

$R_{16} = 61.12\%$

How we implemented equal representation in the training and test set

In order to ensure that there is equal representation of the three classes in the training and test set, we first create a training and test for each class using an 80:20 ratio, respectively. Then, we combine (row bind) the three training sets together and likewise, do the same for the three test sets.

Verify mj/NTST ≃ nj/N for j=1,2,3

j=1

m1/NTST = 0.3080

n1/N = 0.3081

∴ m1/NTST ≃ n1/N

j=2

m2/NTST = 0.3445

n2/N = 0.3446

∴ m2/NTST ≃ n2/N

j=3

m3/NTST = 0.3474

n3/N = 0.3473

∴ m3/NTST ≃ n3/N

Question 2

Geometric representation of Ai in terms of Ei

Ai is the projection of Ei in a 5-dimensional space (previously, we computed a =5, so dim(Ai)=a=5). In other words, Ai is vector in R^5 consisting of the scalar product of Ei in R^{400} and its eigenvalues.

Percentage of successful classifications on TEST and TRAIN

Notation

k = The number of "nearest" neighbors that kNN considers in the model

per(k) = The percentage of correct classifications using k

PerAi.TEST(5) = 0.6381

PerAi.TRAIN (5) = 0.4194

k = 5 correctly classifies 63.81% and 41.94% of the observations in the test and training set into one of the three classes, respectively.

Confusion matrix on TEST and TRAIN

Side Note

COURIER refers to class CL1

CALIBRI refers to class CL2

TIMES refers to class CL3

font.test

knn.5.Ai.test	COURIER	CALIBRI	TIMES
COURIER	467	267	59
CALIBRI	206	492	96
TIMES	179	194	806

font.train

knn.5.Ai.train	COURIER	CALIBRI	TIMES
COURIER	1183	877	829
CALIBRI	1891	2220	1776
TIMES	336	718	1239

Comparing with the results obtained in HW2 for kNN classification with k=5

In HW2, we obtained per(5) = 0.7965 which means k=5 correctly classifies 79.65% of the observations in the test set into one of the three classes. kNN with k=5 performed significantly worse using Ai as the new features of SDATA. Using Ai, we obtained perAi.test (5) = 0.6381

which means k=5 correctly classifies 63.81% of the observations in the test set into one of the three classes. This is about a 16% decrease in the number of observations in the test set that were correctly classified.

Question 3

Geometric representation of Gi in terms of Ei

Gi is the projection of Ei in a 11-dimensional space (wepreviously computed a=5 and b=16, so dim(Gi=b-a=16-5=11). In other words, Gi is vector in R^{11} consisting of the scalar product of Ei in R^{400} and its eigenvalues.

Percentage of successful classifications on TEST and TRAIN

Notation

k = The number of "nearest" neighbors that kNN considers in the model
per(k) = The percentage of correct classifications using k

perGi.TEST(5) = 0.7469
perGi.TRAIN (5) = 0.4804
k = 5 correctly classifies 74.69% and 48.04% of the observations in the test and training set into one of the three classes, respectively.

Confusion matrix on TEST and TRAIN

Side Note

COURIER refers to class CL1
CALIBRI refers to class CL2
TIMES refers to class CL3

```
                font.test
knn.5.Gi.test COURIER CALIBRI TIMES
      COURIER     603     162    89
      CALIBRI      75     623    32
      TIMES       174     168   840
```

```
                font.train
knn.5.Gi.train COURIER CALIBRI TIMES
       COURIER    1100     529   677
       CALIBRI    1644    2755  1705
       TIMES       666     531  1462
```

Comparing with the results obtained in HW2 for kNN classification with k=5

In HW2, we obtained per(5) = 0.7965 which means k=5 correctly classifies 79.65% of the observations into one of the three classes. kNN with k=5 performed slightly worse using Gi as the new features of SDATA. Using Gi, we obtained perGi.test (5) = 0.7469 which means k=5 correctly classifies 74.69% of the observations in the test set into one of the three classes. This is about a 5% decrease in the number of observations in the test set that were correctly classified.

Comparing these results with the preceding results

Overall, applying kNN with k=5 using Gi as the new features of SDATA performed significantly better on the test set and slightly better on the training set than using Ai as the new features. With Gi, we obtained about an 11% and 6% increase in the number of observations in the test and training set were correctly classified into one of the three classes, respectively. Thus, kNN performs better when there are more features in the dataset.

Question 4

Cost function Cost(H1, H2, H3)

The Cost function sums the distortion in the points in the clusters, H1, H2, and H3. Mathematically, it is the sum of squared distances/deviations from the points to their centers and is a measure of the variability of the points within each cluster. A cluster that has a small cost is more compact than a cluster that has a large cost. Clusters that have higher costs exhibit greater variability in distortion between points within the cluster. The K-means algorithm takes the set S of points in the clusters as input and attempts to choose k representatives for S. The distortion on a point $x \in S$ is then the distance to its closest representative. The overall goal of the Cost function is to ensure that every point in S has low distortion. In other words, the function ultimately tries to minimize the maximum distortion in S.

Terminal Costs of Cost(H1, H2, H3)

Notation

N = Current run of K-means algorithm on H1, H2, and H3

$cost_N$(H1, H2, H3) = The sum of the distortion in the points in the clusters, H1, H2, and H3 during run N

$cost_1$(H1, H2, H3) = 1129558.9
$cost_2$(H1, H2, H3) = 966775.9
$cost_3$(H1, H2, H3) = 957697.4
$cost_4$(H1, H2, H3) = 965148.7
$cost_5$(H1, H2, H3) = 1129558.9
$cost_6$(H1, H2, H3) = 966775.9
$cost_7$(H1, H2, H3) = 954011.8

$cost_8(H1, H2, H3) = 1129558.9$
$cost_9(H1, H2, H3) = 962724.7$
$cost_{10}(H1, H2, H3) = 965639.5$

Selecting the clustering result H1 H2 H3 achieving the smallest terminal cost
We achieved the smallest terminal cost during our 7th run of K-means. The cost computed by the 7th run is 954011.8

Question 5

Compute Cost(CL1, CL2, CL3) and compare to Cost(H1, H2, H3)
Notation
N = Current run N of K-means algorithm on CL1, CL2, and CL3
$cost_N(CL1, CL2, CL3)$ = The sum of the distortion in the points in the clusters, CL1, CL2, and CL3 during run N

$cost_1(CL1, CL2, CL3) = 45864651123$
$cost_2(CL1, CL2, CL3) = 45864650957$
$cost_3(CL1, CL2, CL3) = 45864890408$
$cost_4(CL1, CL2, CL3) = 45864677447$
$cost_5(CL1, CL2, CL3) = 45864651123$
$cost_6(CL1, CL2, CL3) = 45864664940$
$cost_7(CL1, CL2, CL3) = 45865045526$
$cost_8(CL1, CL2, CL3) = 45864895221$
$cost_9(CL1, CL2, CL3) = 45864468162$
$cost_{10}(CL1, CL2, CL3) = 45864664940$

Overall, the costs for CL1, CL2, and CL3 are significantly higher than the costs for H1, H2, and H3. Thus, the K-means algorithm computes lower costs when there are fewer features in the dataset.

Compute and interpret the percentages, Pij and Qij
Based on one run of K-means algorithm, we obtained the following percentages.

$P_{ij} = size(H_i \cap CL_j) / size(CL_j)$
$P_{11} = 1$
$P_{12} = 1$
$P_{13} = 1$
$P_{21} = 1$

$P_{22} = 1$
$P_{23} = 1$
$P_{31} = 0.5050$
$P_{32} = 0.4514$
$P_{33} = 0.4480$

During one run of K-means clustering, We obtained the above percentages. P_{ij} is equal to 1 where i=1,2 and j=1,2,3. This means that the computed clusters for H1 and H2 were overestimated. Many observations in the training set for Ai were wrongly assigned to CL1 and CL2 when they should belong to another class. On the other hand, P_{ij} is less than 1 where i=3 and j=1,2,3. This means that the computed cluster for H3 was underestimated and that a fair amount of observations that should belong to CL3 were not accounted for.

Qij = size(Hi ∩ CLj)/ size(Hi)
$Q_{11} = 0.6335$
$Q_{12} = 0.7087$
$Q_{13} = 0.7141$
$Q_{21} = 0.8602$
$Q_{22} = 0.9624$
$Q_{23} = 0.9697$
$Q_{31} = 1$
$Q_{32} = 1$
$Q_{33} = 1$

Using the same run of K-means clustering as above, We obtained the above percentages. Q_{ij} is less than 1 where i=1,2 and j=1,2,3. This means that the sizes for computed clusters for H1 and H2 are larger than the sizes of their actual classes, CL1 and CL2. H1 and H2 includes observations that do not belong to CL1 and CL2, respectively. On the other hand, Q_{ij} is equal 1 where i=3 and j=1,2,3. This means that the size for the computed cluster for H3 is smaller than the size of its actual class, CL3. H3 fails to include all observations that belong to CL3.

Overall, the percentages above show that the predicted cluster sizes, H1, H2, and H3, computed by K-means algorithm differs significantly from their corresponding true clusters sizes, CL1, CL2, and CL3.

 # Code for Part 2
 # Note: Dependent on code from Homework 2

 # Question 1

```
# Find a = smallest integer j such that Rj > 35%
start_time <- Sys.time()
a <- numeric(0)
b <- numeric(0)
for(i in 1:400)
{
  if(Rj[i] > 0.35)
  {
    a <- i
    break
  }
}
a # 5
Rj[a] # 35.02%
# Find b = smallest integer j such that Rj > 60%
for(j in 1:400)
{
  if(Rj[j] > 0.60)
  {
    b <- j
    break
  }
}
b # 16
Rj[b] # 61.12%
splitCL1 <- floor(nrow(CL1)*.20)
testSizeCL1 <- 1:splitCL1
train.CL1 <- CL1[-testSizeCL1,]
test.CL1 <- CL1[testSizeCL1,]
m1 <- nrow(test.CL1)
# Training and test set for CL2
splitCL2 <- floor(nrow(CL2)*.20)
testSizeCL2 <- 1:splitCL2
train.CL2 <- CL2[-testSizeCL2,]
test.CL2 <- CL2[testSizeCL2,]
```

```
m2 <- nrow(test.CL2)
# Training and test set for CL3
splitCL3 <- floor(nrow(CL3)*.20)
testSizeCL3 <- 1:splitCL3
train.CL3 <- CL3[-testSizeCL3,]
test.CL3 <- CL3[testSizeCL3,]
m3 <- nrow(test.CL3)
# Combine training sets to form SDATA training set
SDATA.train1 <- rbind(train.CL1, train.CL2)
SDATA.train2 <- rbind(SDATA.train1, train.CL3)
SDATA.train2$font <- factor(SDATA.train2$font)
SDATA.train <- SDATA.train2[,4:403]
font.train <- SDATA.train2[,1]
# Combine test sets to form SDATA test set
SDATA.test1 <- rbind(test.CL1, test.CL2)
SDATA.test2 <- rbind(SDATA.test1, test.CL3)
SDATA.test2$font <- factor(SDATA.test2$font)
SDATA.test <- SDATA.test2[,4:403]
font.test <- SDATA.test2[,1]
NTST <- nrow(SDATA.test)
knn.5 <- knn(SDATA.train, SDATA.test, font.train, k=5)
table(knn.5, font.test)
# Verify mj/NTST ≈ nj/N for j=1,2,3
# j=1
m1/NTST # 0.3080
n1/N # 0.3081
# j=2
m2/NTST # 0.3445
n2/N # 0.3446
# j=3
m3/NTST # 0.3474
n3/N # 0.3473
end_time <- Sys.time()
end_time - start_time # # Computation time: 0.48 secs
```

 # Question 2

```
# Create vector Ai in R^5
# dim(Ai) = a = 5
# Calculate the first five scores
```

```
start_time <- Sys.time()
scor1 <- numeric(0)
scor2 <- numeric(0)
scor3 <- numeric(0)
scor4 <- numeric(0)
scor5 <- numeric(0)
for (i in 1:ncol(TSDATA))
{
  scor1[i] = TSDATA[,i] %*% eigenvectors[,1]
  scor2[i] = TSDATA[,i] %*% eigenvectors[,2]
  scor3[i] = TSDATA[,i] %*% eigenvectors[,3]
  scor4[i] = TSDATA[,i] %*% eigenvectors[,4]
  scor5[i] = TSDATA[,i] %*% eigenvectors[,5]
}
Ai <- do.call("cbind", list(scor1, scor2, scor3, scor4, scor5))
# Perform k=5 using Ai
set.seed(1)
library(class)
Ai.test <- do.call("rbind", list(Ai[1:852,], Ai[4263:5215,], Ai[9031:9991,]))
Ai.train <- do.call("rbind", list(Ai[853:4262,], Ai[5216:9030,], Ai[9992:13835,]))
knn.5.Ai.test <- knn(Ai.train, Ai.test, font.train, k=5)
table(knn.5.Ai.test, font.test)
per.5.Ai.test <- round(((467+492+806)/2766), 4)
per.5.Ai.test # 0.6381
knn.5.Ai.train <- knn(Ai.test, Ai.train, font.test, k=5)
table(knn.5.Ai.train, font.train)
per.5.Ai.train <- round(((1183+2220+1239)/11069),4)
per.5.Ai.train # 0.4194
end_time <- Sys.time()
end_time - start_time # Computation time: 1.74 secs
```

 # Question 3

```
# Create vector Gi in R^11
# dim(Gi) = b-a = 16-5 = 11
# Calculate scores 6-16
start_time <- Sys.time()
scor6 <- numeric(0)
scor7 <- numeric(0)
scor8 <- numeric(0)
```

```
scor9 <- numeric(0)
scor10 <- numeric(0)
scor11 <- numeric(0)
scor12 <- numeric(0)
scor13 <- numeric(0)
scor14 <- numeric(0)
scor15 <- numeric(0)
scor16 <- numeric(0)
for (i in 1:ncol(TSDATA))
{
  scor6[i] = TSDATA[,i] %*% eigenvectors[,6]
  scor7[i] = TSDATA[,i] %*% eigenvectors[,7]
  scor8[i] = TSDATA[,i] %*% eigenvectors[,8]
  scor9[i] = TSDATA[,i] %*% eigenvectors[,9]
  scor10[i] = TSDATA[,i] %*% eigenvectors[,10]
  scor11[i] = TSDATA[,i] %*% eigenvectors[,11]
  scor12[i] = TSDATA[,i] %*% eigenvectors[,12]
  scor13[i] = TSDATA[,i] %*% eigenvectors[,13]
  scor14[i] = TSDATA[,i] %*% eigenvectors[,14]
  scor15[i] = TSDATA[,i] %*% eigenvectors[,15]
  scor16[i] = TSDATA[,i] %*% eigenvectors[,16]
}
Gi <- do.call("cbind", list(scor6, scor7, scor8, scor9, scor10, scor11,
                        scor12, scor13, scor14, scor15, scor16))
# Perform k=5 using Gi
set.seed(1)
Gi.test <- do.call("rbind", list(Gi[1:852,], Gi[4263:5215,], Gi[9031:9991,]))
Gi.train <- do.call("rbind", list(Gi[853:4262,], Gi[5216:9030,], Gi[9992:13835,]))
knn.5.Gi.test <- knn(Gi.train, Gi.test, font.train, k=5)
table(knn.5.Gi.test, font.test)
per.5.Gi.test <- round((603+623+840)/2766, 4)
per.5.Gi.test # 0.7469
knn.5.Gi.train <- knn(Gi.test, Gi.train, font.test, k=5)
table(knn.5.Gi.train, font.train)
per.5.Gi.train <- round((1100+2755+1462)/11069, 4)
per.5.Gi.train # 0.4804
end_time <- Sys.time()
end_time - start_time # Computation time: 2.30 secs
```

 # Question 4

```
# Create vector of 10 different random centers from 1 to 20
start_time <- Sys.time()
# Run K-means 10 times on H1 H2 H3 (Ai.train) with a different center
# and compute its cost
costs <- numeric(0)
smallestCost <- 0
smallest.i <- 0
for(i in 1:10)
{
 set.seed(i)
 Ai.kmeans <- kmeans(Ai.train, 3)
 H1 <- Ai.train[which(Ai.kmeans$cluster==1),]
 H2 <- Ai.train[which(Ai.kmeans$cluster==2),]
 H3 <- Ai.train[which(Ai.kmeans$cluster==3),]
 H1.center <- Ai.kmeans$centers[1,]
 H2.center <- Ai.kmeans$centers[2,]
 H3.center <- Ai.kmeans$centers[3,]
 # Calculate the differences for each cluster
 H1.diff <- numeric(0)
 for(a in 1:nrow(H1))
 {
  H1.diff[a] <- sum(H1[a,]-H1.center)
 }
 H2.diff <- numeric(0)
 for(b in 1:nrow(H2))
 {
  H2.diff[b] <- sum(H2[b,]-H2.center)
 }
 H3.diff <- numeric(0)
 for(c in 1:nrow(H3))
 {
  H3.diff[c] <- sum(H3[c,]-H3.center)
 }
 # Calculate the cost by adding the 3 differences
 costs[i] <- sum(H1.diff^2) + sum(H2.diff^2) + sum(H3.diff^2)
 # Find the smallest cost
 if(smallest.i == 0 || smallestCost > costs[i])
 {
```

```
   smallest.i <- i
   smallestCost <- costs[i]
  }
}
costs # [1129558.9, 966775.9, 957697.4, 965148.7, 1129558.9,
     #  962724.7, 965639.5, 966775.9, 954011.8, 1129558.9]
smallest.i # Implementation 7
end_time <- Sys.time()
end_time - start_time # Computation time: 0.46 secs

# Question 5
# Run k-Means 10 times on CL1 CL2 CL3 (SDATA.train) with a different center
# and compute its cost
start_time <- Sys.time()
costs2 <- numeric(0)
for(i in 1:10)
{
  set.seed(i)
  kmeans.SDATA <- kmeans(SDATA.train, 3)
  # Cost function = total sum of squared differences
  # In this case, We use tot.withinss since calculating
  # using the formula takes too much computation time
  costs2[i] <- kmeans.SDATA$tot.withinss
}
costs2 # [45864651123, 45864650957, 45864890408, 45864677447, 45864651123,
       # ,45864664940, 45865045526, 45864895221, 45864468162, 45864664940]
# Compute all the percentages
set.seed(1)
km.Ai <- kmeans(Ai.train, 3)
# Computed cluster sizes for H1, H2, and H3
km.Ai$size # Cluster sizes: 5383, 3964, 1722
H1.size <- 5383
H1 <- 1:H1.size
H2.size <- 3964
H2 <- 1:H2.size
H3.size <- 1722
H3 <- 1:H3.size
# "Ideal" clustering CL1, CL2, CL3 (using their respective training sets)
CL1.size <- nrow(train.CL1) # 3410
```

```
CL.1 <- 1:CL1.size
CL2.size <- nrow(train.CL2) # 3815
CL.2 <- 1:CL2.size
CL3.size <- nrow(train.CL3) # 3844
CL.3 <- 1:CL3.size
# Pij = size(Hi∩CLj)/ size(CLj)
P <- matrix(1:9, nrow = 3, ncol = 3)
P[1,1] <- length(intersect(H1, CL.1))/CL1.size
P[1,1] # 1
P[1,2] <- length(intersect(H1, CL.2))/CL2.size
P[1,2] # 1
P[1,3] <- length(intersect(H1, CL.3))/CL3.size
P[1,3] # 1
P[2,1] <- length(intersect(H2, CL.1))/CL1.size
P[2,1] # 1
P[2,2] <- length(intersect(H2, CL.2))/CL2.size
P[2,2] # 1
P[2,3] <- length(intersect(H2, CL.3))/CL3.size
P[2,3] # 1
P[3,1] <- length(intersect(H3, CL.1))/CL1.size
P[3,1] # 0.5050
P[3,2] <- length(intersect(H3, CL.2))/CL2.size
P[3,2] # 0.4514
P[3,3] <- length(intersect(H3, CL.3))/CL3.size
P[3,3] # 0.4480
# Qij = size(HCLj)/ size(Hi)
Q <- matrix(1:9, nrow = 3, ncol = 3)
Q[1,1] <- length(intersect(H1, CL.1))/H1.size
Q[1,1] # 0.6335
Q[1,2] <- length(intersect(H1, CL.2))/H1.size
Q[1,2] # 0.7087
Q[1,3] <- length(intersect(H1, CL.3))/H1.size
Q[1,3] # 0.7141
Q[2,1] <- length(intersect(H2, CL.1))/H2.size
Q[2,1] # 0.8602
Q[2,2] <- length(intersect(H2, CL.2))/H2.size
Q[2,2] # 0.9624
Q[2,3] <- length(intersect(H2, CL.3))/H2.size
Q[2,3] # 0.9697
```

```
Q[3,1] <- length(intersect(H3, CL.1))/H3.size
Q[3,1] # 1
Q[3,2] <- length(intersect(H3, CL.2))/H3.size
Q[3,2] # 1
Q[3,3] <- length(intersect(H3, CL.3))/H3.size
Q[3,3] # 1
end_time <- Sys.time()
end_time - start_time # Computation time: 15.50 secs
```