Anna Hathaway

STAT:4540 Project Part 2

**1.**

i. The first PC is is a set of X1, X2, …, Xp which is a normalized linear combination with the largest variance. The normalization refers to the loading vectors in the problem. We have assumed that the variables in our matrix X have been centered around zero and standard deviation of 1. Hence the optimization problem we are solving is maximizing the sample variance of n values. The second PC component is the linear combination of X1… Xp that has maximum variance out of linear combination not correlated with Z1.The second loading vector is orthogonal to the first loading vector. In this problem, the first PC score vectors have length n=1427 and the PC loading vectors have length p=16.

ii.

sdev= standard deviation, rotation= PC loadings, center= mean of variables used for scaling before implementing PCA or (mean), scale= standard deviation of variables used for scaling before implementing PCA or sqrt(var)

iii.

Center and scale are used in data pre-processing because they are the mean and standard deviation of the variables used for scaling before implementing PCA.

iv.

The two loading vectors are related to rotation because each column of the rotation matrix provides the corresponding PC loading vector.

v.

The geometric interpretations of the loading vectors relate to the linear algebra ordered sequence of eigenvectors and the variances of the components are eigenvalues. On a graph, the vectors represent the axis and correlation between the variables.

vi.

φ1 defines a direction on which the data varies most. When x1 is projected on φ1 then z1 is obtained. φ2 defines a direction on which the data varies most after accounting for φ1.

vii.

The standard deviation after PCA of the first row of x is 2.0876073. Similarly in the second row of x, the standard deviation is 1.7845993.

viii.

I suggest increasing the number of principal components because the first PC only explains 27.3% of the variance and the second PC only explains 19.9% of the variance.

**2.**

i.-ii. see attached code

iii.

table(comp.summ.1[ , 2])

253/(131 +253 +364 +332) = 0.2342593

table(comp.summ.2[ , 2])

111/(232 +111 +4) = 0.3198847

table(avg.summ.1[ , 2])

364/(364+364+336) = 0.3421053

table(avg.summ.2[ , 2])

0/363

table(sgl.summ.1[ , 2])

364/(363 +364 +364 +335) = 0.2552595

table(sgl.summ.2[ , 2])

iv. Order from lowest to high: complete, single, then average based on their purities calculated above. Complete is better than the rest.

v. Performing the same clustering steps in. just a subset of the data improves the size and purity of the clusters because in general performing clustering on just the first few principal component score vectors gives better results than performing clustering on the full data set.

**3.**

i.

The 2-means clustering optimization partitions the observations into 2 disjoint sets. The objective is to minimize the amount by which all the observations within a cluster differ from each other. We are minimizing the sum from k=1 to K of the within cluster variation. The dissimilarity measure is a numerical value that will measure how different two data objects are inside of the cluster. 0 is objects are alike while infinity is objects are very different.

ii.

We require 10 random initializations of the cluster centers because we need an initialization larger than one. We use this in the first step of the algorithm for k-means clustering. 10 is large enough to work without getting too large.

Cluster centers:

Cluster means:

dim1 dim2 dim3 dim4 dim5 dim6 dim7

1 34.88824 88.35588 9.773529 59.49412 11.97059 19.95882 48.78235

2 21.48298 74.69457 48.429623 93.91996 70.16007 85.19319 58.66881

dim8 dim9 dim10 dim11 dim12 dim13 dim14

1 5.941176 86.38824 28.68529 93.18529 67.48824 62.56765 90.988235

2 59.938362 51.89604 35.81233 45.82889 16.91260 48.96228 7.105796

dim15 dim16

1 24.18235 79.55294

2 57.41122 6.25575

iii.

table(km.summ.1[ , 2])

0/340

#[1] 0

table(km.summ.2[ , 2])

364/(23+ 364+ 364+ 336)

#[1] 0.3348666

iv.

Yes, this approach improves the size and purities of the clusters because in general performing clustering on just the first few principal component score vectors gives better results than performing clustering on the full data set.

**4.**

i.

> table(test$class, pred.yhat)

pred.yhat

0 1

0 211 2

1 9 64

ii.

> table(test$class, pred.yhat)

pred.yhat

0 1

0 210 3

1 10 63

#Ridge classification and lasso classification give very similar results.

R CODE

setwd("~/Downloads")

pendigits <- read.csv("pendigits\_subset.csv")

#Question 1

#i.

x <- as.matrix(pendigits[ , 1:16])

dim(x)

#ii.

pcout <- prcomp(x, scale=TRUE)

names(pcout)

#iii.

pcout$center

pcout$scale

#iv.

pcout$rotation

#vii.

pcout$sdev

pcout$x

#viii.

pcvar <- pcout$sdev^2

pve <- pcvar/sum(pcvar)

pve

#Question 2

#i.

hc.comp <- hclust(dist(x), method="complete")

hc.avg <- hclust(dist(x), method ="average")

hc.sgl <- hclust(dist(x), method ="single")

#ii.

clust.comp = cutree(hc.comp, 2)

clust.avg = cutree(hc.avg, 2)

clust.sgl = cutree(hc.sgl, 2)

#iii.

comp.summ = cbind(comp = clust.comp, true = pendigits$class)

avg.summ = cbind(avg = clust.avg, true = pendigits$class)

sgl.summ = cbind(sgl = clust.sgl, true = pendigits$class)

comp.summ.1 = comp.summ[comp.summ[ , 1] == 1, ]

comp.summ.2 = comp.summ[comp.summ[ , 1] == 2, ]

table(comp.summ.1[ , 2])

253/(131 +253 +364 +332)

#result [1] 0.2342593

table(comp.summ.2[ , 2])

111/(232 +111 +4)

#result [1] 0.3198847

avg.summ.1 = avg.summ[avg.summ[ , 1] == 1, ]

avg.summ.2 = avg.summ[avg.summ[ , 1] == 2, ]

table(avg.summ.1[ , 2])

364/(364+364+336)

#result [1] 0.3421053

table(avg.summ.2[ , 2])

0/363

sgl.summ.1 = sgl.summ[sgl.summ[ , 1] == 1, ]

sgl.summ.2 = sgl.summ[sgl.summ[ , 1] == 2, ]

table(sgl.summ.1[ , 2])

364/(363 +364 +364 +335 )

#result [1] 0.2552595

table(sgl.summ.2[ , 2])

#iv.ordered low to high

#v.

hc.comp2 <- hclust(dist(pcout$x[ , 1:2]), method="complete")

hc.avg2 <- hclust(dist(pcout$x[ , 1:2]), method ="average")

hc.sgl2 <- hclust(dist(pcout$x[ , 1:2]), method ="single")

clust.comp2 = cutree(hc.comp2, 2)

clust.avg2 = cutree(hc.avg2, 2)

clust.sgl2 = cutree(hc.sgl2, 2)

comp.summ2 = cbind(comp = clust.comp2, true = pendigits$class)

avg.summ2 = cbind(avg = clust.avg2, true = pendigits$class)

sgl.summ2 = cbind(sgl = clust.sgl2, true = pendigits$class)

comp.summ.1.2 = comp.summ2[comp.summ2[ , 1] == 1, ]

table(comp.summ.1.2[ , 2])

228/(228+849)

#result [1] 0.2116992

comp.summ.2.2 = comp.summ2[comp.summ2[ , 1] == 2, ]

table(comp.summ.2.2[ , 2])

136/(136+214)

#result [1] 0.3885714

avg.summ.1.2 = avg.summ2[avg.summ2[ , 1] == 1, ]

avg.summ.2.2 = avg.summ2[avg.summ2[ , 1] == 2, ]

table(avg.summ.1.2[ , 2])

364/(364+712)

#result [1] 0.33829

table(avg.summ.2.2[ , 2])

0/351

sgl.summ.1.2 = sgl.summ2[sgl.summ2[ , 1] == 1, ]

sgl.summ.2.2 = sgl.summ2[sgl.summ2[ , 1] == 2, ]

table(sgl.summ.1.2[ , 2])

table(sgl.summ.2.2[ , 2])

#Question 3

#i. described optimization problem

#ii.

km.out <- kmeans (x, 2, nstart =10)

km.out

km <- km.out$cluster

#iii.

km.summ = cbind(comp = km, true = pendigits$class)

km.summ.1 = km.summ[km.summ[ , 1] == 1, ]

km.summ.2 = km.summ[km.summ[ , 1] == 2, ]

table(km.summ.1[ , 2])

0/340

#[1] 0

table(km.summ.2[ , 2])

364/(23+ 364+ 364+ 336)

#[1] 0.3348666

#iv.

km2 <- pcout$x[ , 1:2]

km.summ.2 = cbind(comp = km2, true = pendigits$class)

km.summ.1.2 = km.summ.2[km.summ.2[ , 1] == 1, ]

km.summ.2.2 = km.summ.2[km.summ.2[ , 1] == 2, ]

table(km.summ.1[ , 2])

table(km.summ.2[ , 2])

#Question 4

pendigits$class = as.numeric(pendigits$class == 1)

set.seed(1)

#training data

idx <- sample(1:nrow(pendigits), 0.80 \* nrow(pendigits))

#test data

idx0 <- setdiff(1:nrow(pendigits), idx)

#

train = pendigits[idx, ]

test = pendigits[idx0, ]

#i. ridge classification

train.mat <- model.matrix(class~ ., data=train)

test.mat <- model.matrix(class~ ., data=test)

library(glmnet)

grid <- 10^seq(4, -2, length=100)

fit.ridge <- glmnet(train.mat, train$class, alpha=0, lambda=grid, family = "binomial")

cv.ridge = cv.glmnet(train.mat, train$class, alpha=0, lambda=grid, family = "binomial")

bestlam.ridge = cv.ridge$lambda.min

bestlam.ridge

pred.ridge <- predict(fit.ridge, s=bestlam.ridge, newx=test.mat, type = "response")

pred.yhat = as.numeric(pred.ridge > 0.5)

table(test$class, pred.yhat)

#ii. lasso classification

fit.lasso <- glmnet(train.mat, train$class, alpha=1, lambda=grid, thresh=1e-12, family = "binomial")

cv.lasso <- cv.glmnet(train.mat, train$class, alpha=1, lambda=grid, thresh=1e-12, family = "binomial")

bestlam.lasso <- cv.lasso$lambda.min

bestlam.lasso

#[1] 0.01

pred.lasso <- predict(fit.lasso, s=bestlam.lasso, newx=test.mat, type = "response")

pred.yhat = as.numeric(pred.lasso > 0.5)

table(test$class, pred.yhat)