CS 498 AML HW3 Report

Author: Moyu Liu

Dataset: https://www.kaggle.com/c/aml-hw3
Language: R-Studio

50 MSE Results

0N	1N	2N	3N	4N
4.54247066666666	0.383450311504984	0.175563000244339	0.14178364800457	0.160838361807638
4.54247066666666	0.641093184900985	0.715628487504956	0.908392907398274	1.11565785784931
4.54247066666666	1.2903724507598	1.96724039237987	2.65084113513274	3.65327973251111
4.54247066666666	0.799942743733825	0.828082554706743	0.828082554706743	1.194
4.54247066666666	1.91776774994606	3.33172210394032	4.54825719724982	5.13926666666667

0c	1c	2c	3c	4c
4.54311902907455	0.384613533957617	0.177815282669625	0.144440506031377	0.160838361807638
4.54953899271544	0.648642108410852	0.750621128999984	0.941972819285056	1.11565785784931
4.55747296393055	1.32346214804188	2.11974804928195	3.02737991997533	3.65327973251111
4.56619866666667	0.840614157257198	1.20708979682591	1.27119196718607	1.194
4.919928	2.83567942802644	4.6514345027171	4.9712472715256	5.13926666666667

Read CSV files

```r
iris = read.csv("dataset/iris.csv")
data1 = read.csv("dataset/dataI.csv")
data2 = read.csv("dataset/dataII.csv")
data3 = read.csv("dataset/dataIII.csv")
data4 = read.csv("dataset/dataIV.csv")
data5 = read.csv("dataset/dataV.csv")

mse = data.frame(matrix(nrow = 5, ncol = 10))
colnames(mse)[colnames(mse)=="X1"] <- '0N'
colnames(mse)[colnames(mse)=="X2"] <- '1N'
colnames(mse)[colnames(mse)=="X3"] <- '2N'
colnames(mse)[colnames(mse)=="X4"] <- '3N'
colnames(mse)[colnames(mse)=="X5"] <- '4N'
colnames(mse)[colnames(mse)=="X6"] <- '0c'
colnames(mse)[colnames(mse)=="X7"] <- '1c'
colnames(mse)[colnames(mse)=="X8"] <- '2c'
colnames(mse)[colnames(mse)=="X9"] <- '3c'
colnames(mse)[colnames(mse)=="X10"] <- '4c'

cov_iris = cov(iris)
mean_iris = c(0,0,0,0)
for(i in 1:4) {
  mean_iris[i] = mean(iris[,i])
}

eig_vec = eigen(cov_iris)$vectors
```

Calculate PCA and MSE for mea

```r
pca_func2 = function(data, table_num) {

  row_num = nrow(data)
  col_num = ncol(data)

  m = c(0,0,0,0)
  for(i in 1:4) {
    m[i] = mean(data[,i])
  }

  eig_vec = eigen(cov(data))$vectors

  ret_data = matrix(rep(0, row_num * col_num), nrow = row_num, ncol = col_num)
  if(table_num == 0) {
    for(i in 1:ncol(data)) {
      ret_data[,i] = m[i]
    }
  }
  else {
    for(i in 1:row_num) {
      for(j in 1:table_num) {
        ret_data[i,] = ret_data[i,] + as.numeric(eig_vec[,j] %*% (data[i,] - m)) * eig_vec[,j]
      }
      ret_data[i,] = ret_data[i,] + m
    }
  }

  square_diff = (ret_data - iris)^2
  err = 0
  for(i in 1:row_num) {
    for(j in 1:col_num) {
      err = err + (ret_data[i,j] - iris[i,j])^2
    }
  }

  err = err / nrow(data)

  return(err)
}
```

Construct MSE table

```r
mse[1,1] = pca_func(data1, 0)
mse[2,1] = pca_func(data2, 0)
mse[3,1] = pca_func(data3, 0)
mse[4,1] = pca_func(data4, 0)
mse[5,1] = pca_func(data5, 0)

mse[1,2] = pca_func(as.matrix(data1), 1)
mse[2,2] = pca_func(as.matrix(data2), 1)
mse[3,2] = pca_func(as.matrix(data3), 1)
mse[4,2] = pca_func(as.matrix(data4), 1)
mse[5,2] = pca_func(as.matrix(data5), 1)

mse[1,3] = pca_func(as.matrix(data1), 2)
mse[2,3] = pca_func(as.matrix(data2), 2)
mse[3,3] = pca_func(as.matrix(data3), 2)
mse[4,3] = pca_func(as.matrix(data4), 2)
mse[5,3] = pca_func(as.matrix(data5), 2)

mse[1,4] = pca_func(as.matrix(data1), 3)
mse[2,4] = pca_func(as.matrix(data2), 3)
mse[3,4] = pca_func(as.matrix(data3), 3)
mse[4,4] = pca_func(as.matrix(data4), 2)
mse[5,4] = pca_func(as.matrix(data5), 3)

mse[1,5] = pca_func(as.matrix(data1), 4)
mse[2,5] = pca_func(as.matrix(data2), 4)
mse[3,5] = pca_func(as.matrix(data3), 4)
mse[4,5] = pca_func(as.matrix(data4), 4)
mse[5,5] = pca_func(as.matrix(data5), 4)
```

```r
pca_func = function(data, table_num) {

  row_num = nrow(data)
  col_num = ncol(data)

  ret_data = matrix(rep(0, row_num * col_num), nrow = row_num, ncol = col_num)
  if(table_num == 0) {
    for(i in 1:ncol(data)) {
      ret_data[,i] = mean_iris[i]
    }
  }
  else {
    for(i in 1:row_num) {
      for(j in 1:table_num) {
        ret_data[i,] = ret_data[i,] + as.numeric(eig_vec[,j] %*% (data[i,] - mean_iris)) * eig_vec[,j]
      }
      ret_data[i,] = ret_data[i,] + mean_iris
    }
  }

  square_diff = (ret_data - iris)^2
  err = 0
  for(i in 1:row_num) {
    for(j in 1:col_num) {
      err = err + (ret_data[i,j] - iris[i,j])^2
    }
  }

  err = err / nrow(data)
  return(err)
}
```