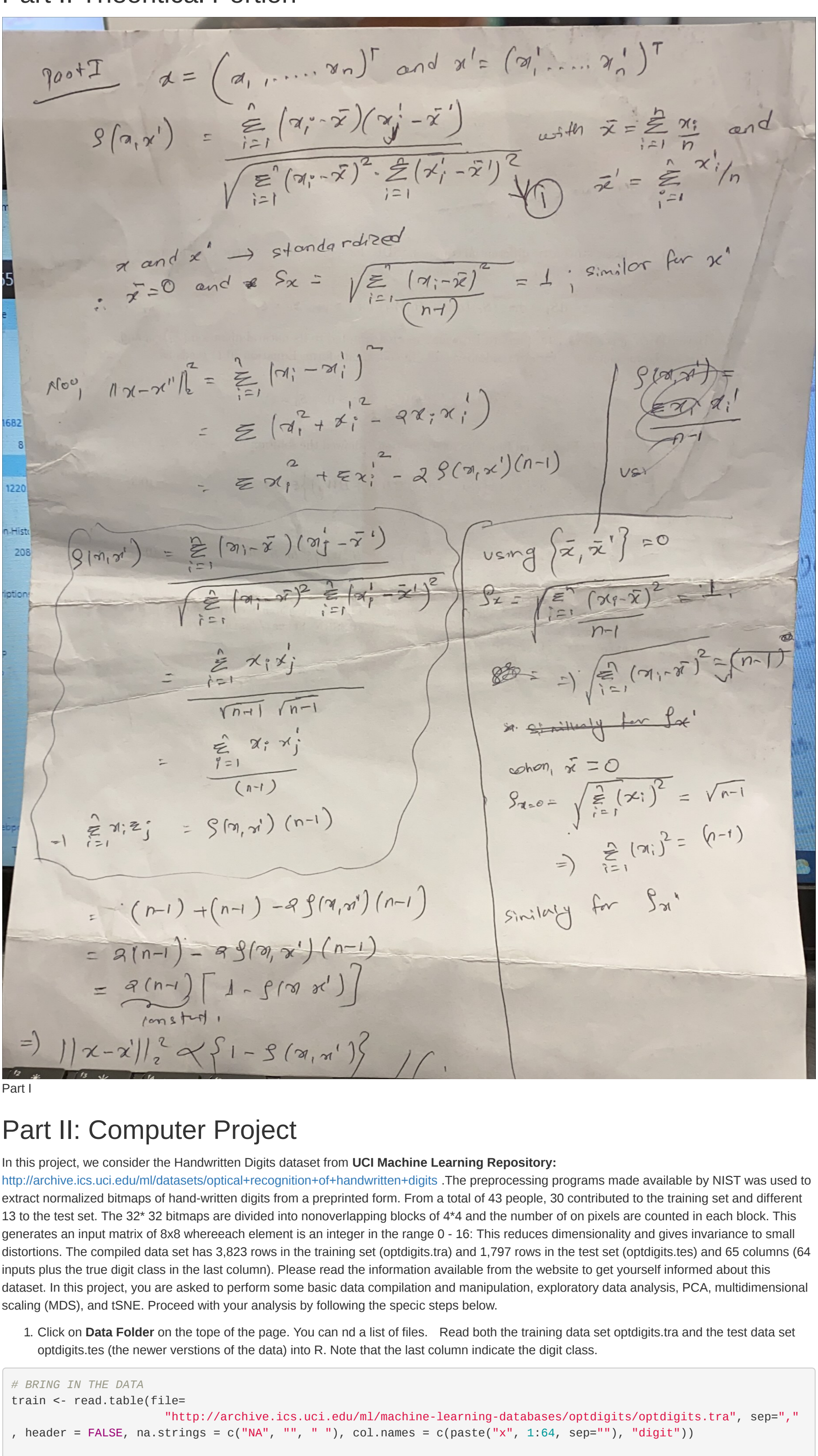
Project-2

Chitra karki

9/27/2021

Part I: Theoritical Portion



Part I

Part II: Computer Project

In this project, we consider the Handwritten Digits dataset from **UCI Machine Learning Repository**: http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits. The preprocessing programs made available by NIST were used to extract normalized bitmaps of hand-written digits from a preprinted form. From a total of 43 people, 30 contributed to the training set and different 13 to the test set. The 32×32 bitmaps are divided into nonoverlapping blocks of 4×4 and the number of on pixels are counted in each block. This generates an input matrix of 8×8 where each element is an integer in the range 0..16. This reduces dimensionality and gives invariance to small distortions. The compiled data set has 3,823 rows in the training set (optdigits.tra) and 1,797 rows in the test set (optdigits.tes) and 65 columns (64 inputs plus the true digit class in the last column). Please read the information available from the website to get yourself informed about this dataset. In this project, you are asked to perform some basic data compilation and visualization, manipulation, exploratory data analysis, PCA, multidimensional scaling (MDS), and tSNE. Proceed with your analysis by following the specs/ steps below.

1. Click on **Data Folder** on the top of the page. You can nd a list of les. Read both the training data set optdigits.tra and the test data set optdigits.tes into (for newer versions of the data) into R. Note that the last column indicate the digit class.

```
# getting to the data
train <- read.table(file=
  "http://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/optdigits.tra", sep=",", header = FALSE, na.strings = c("NA", "", " "), col.names = c(paste("x", 1:64, sep=""), "digit"))

test <- read.table(file="http://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/optdigits.tes", sep=",", header = FALSE, na.strings = c("NA", "", " "), col.names = c(paste("x", 1:64, sep=""), "digit"))

dim(train); dim(test)
```

```
## [1] 3823   65
```

```
## [1] 1797   65
```

Concatenate both data sets into one, call it dat.

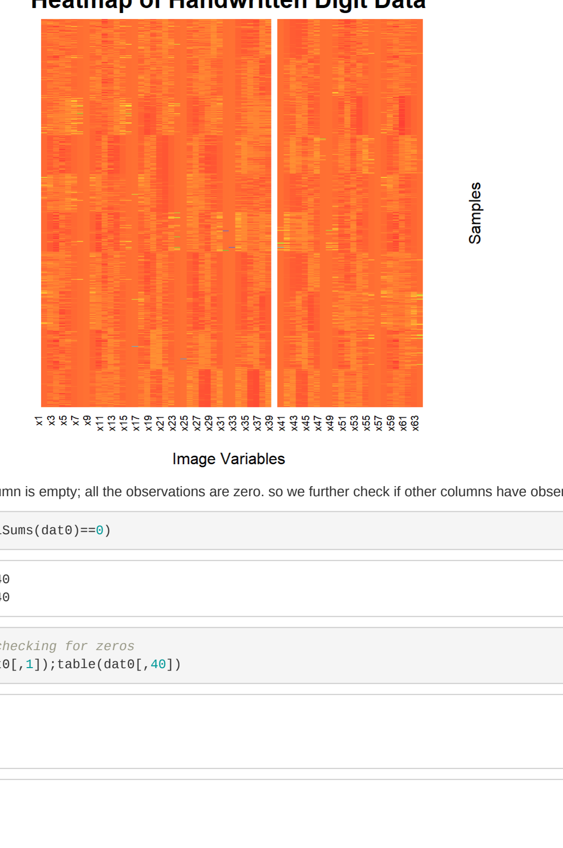
```
dat = rbind(train, test)
dim(dat)
```

```
## [1] 5620   65
```

```
col.names = c(paste("x", 1:64, sep=""), "digit")
```

2. Perform Exploratory Data Analysis (EDA) on dat. While many EDA tools are applicable, let's try out a graphical heatmap presentation of the data. First sort the data according to the digit class so that those for each digits are piled together. Then obtain a heat map of the data (excluding the last column digit) and comment on any interesting ndings. In particular, if there are columns or variables with unary values, you might want to remove them in the ensuing analyses.

```
dat0 <- data.matrix(dat[order(dat$digit), -65])
colnames(dat0) = c(paste("x", 1:64, sep=""))
n <- NROW(dat0)
color <- rainbow(n, alpha = 0.8)
heatmap(dat0, col=color, scale="column", Rowv=NA, Colv=NA,
        labRow=FALSE, margins=c(4,4), xlab="Image Variables", ylab="Samples",main="Heatmap of Handwritten Digit Data")
```



The x40 column is empty, all the observations are zero, so we further check if other columns have observations zero.

```
which(colSums(dat0)==0)
```

```
## x1 x40
##  1  40
```

```
#double checking for col1
table(dat0[,1]);table(dat0[,40])
```

```
##
##    0
## 5620
```

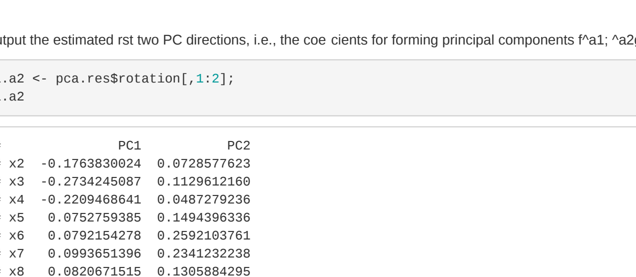
```
##
##    0
## 5620
```

x1 and x40 has column = 0. meaning that all the observations are zero, so we are dropping these columns. And the plot is shown below.

```
# getting after droping 1 and 40 column
heatmap(dat0[,c(-1,-40)], col=color, scale="column", Rowv=NA, Colv=NA,
        labRow=FALSE, margins=c(4,4), xlab="Image Variables", ylab="Samples",main="Heatmap of Handwritten Digit Data")
```

3. Principal Components Analysis (PCA). Run PCA with with dat0 (i.e., with the column digit excluded) and obtain the scree plot showing the cumulative proportions of variation explained by the rst leading PCs.

```
[illegible]
```

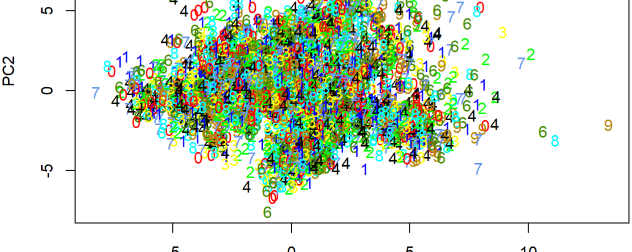


Output the estimated rst two PC directions, i.e., the rotation matrix for forming principal components PC1 and PC2.

```
## x2  [illegible]
## x3  [illegible]
## x4  [illegible]
## x5   0.0752759385  0.1494396336
## x6   0.0792154278  0.2592103761
## x7   0.0993651396  0.2341232238
## x8   0.0820671515  0.1305884295
## x9  -0.0166136180  0.0009636991
## x10 -0.2316687445  0.0597010278
## x11 -0.2344257865  0.0476143709
## x12  0.0487205172 -0.1107548574
## x13 -0.0323696309  0.0448630038
## x14  0.0106011572  0.2339568511
## x15  0.1339823614  0.2540618708
## x16  0.1106767724  0.1227564498
## x17 -0.0072261532  0.0012173433
## x18 -0.1307089266 -0.0110976294
## x19  0.0139352561 -0.1498939354
## x20  0.1028141044 -0.1303848680
## x21 -0.1350793517  0.0896619720
## x22  0.0513328096  0.1763545618
## x23  0.1982744014  0.1326815292
## x24  0.1104294319  0.0477815762
## x25  0.0011422714 -0.0034131301
## x26  0.0935003006 -0.0885620515
## x27  0.1536241765 -0.1622675852
## x28 -0.0341370518  0.0626460533
## x29 -0.1205278945  0.2073896146
## x30  0.1510813447  0.1411890225
## x31  0.1990106156  0.0286887778
## x32  0.0520138954 -0.0002946510
## x33  0.0425264003 -0.0094125674
## x34  0.2070957086 -0.1555447063
## x35  0.2041037004 -0.1594890006
## x36  0.0106290737  0.0582146262
## x37  0.0160802055  0.1188153939
## x38  0.1622148659  0.0169835695
## x39  0.1164965316 -0.0516177078
## x41  0.0749821530 -0.0363819281
## x42  0.1660570698 -0.1481337573
## x43  0.0973822629 -0.2244268078
## x44  0.0319245378 -0.0181996524
## x45  0.1393003227  0.0764626672
## x46  0.0990177042 -0.1080810277
## x47 -0.0498658377 -0.1991298601
## x48 -0.0070638388 -0.0722234632
## x49  0.0458057204 -0.0268616298
## x50 -0.0358002125 -0.0494186961
## x51 -0.1600216621 -0.1406474242
## x52 -0.0565329685 -0.0222883993
## x53  0.0856499249 -0.0084724788
## x54 -0.0963534345 -0.1726462188
## x55 -0.1546803893 -0.2104566855
## x56 -0.0512583389 -0.1063358310
## x57  0.0006071096 -0.0008331468
## x58 -0.1542220769  0.0625119548
## x59 -0.2601618391  0.1380498529
## x60 -0.1952201615  0.0354772522
## x61 -0.0588716253 -0.2449018435
## x62 -0.1423460486 -0.2051089694
## x63 -0.1544929093 -0.1361040639
## x64 -0.0779169359 -0.0542957373
```

Plot PC2 vs. PC1 with a scatterplot, where the `dots' for each digit are represented with di erent colors and digit symbols. This corresponds to the classical MDS analysis. Do you see any pattern? Interpret your results.

```
colr = rep(NA,nrow(dat))
colr[which(dat$digit==0)]="red"
colr[which(dat$digit==1)]="blue"
colr[which(dat$digit==2)]="green"
colr[which(dat$digit==3)]="yellow"
colr[which(dat$digit==4)]="gray"
colr[which(dat$digit==4)]="black"
colr[which(dat$digit==6)]="chartreuse4"
colr[which(dat$digit==7)]="cornflowerblue"
colr[which(dat$digit==8)]="cyan"
colr[which(dat$digit==9)]="darkgoldenrod"
plot(pca.res$x[,c(1,2)],pch=as.character(dat$digit),col=colr)
```



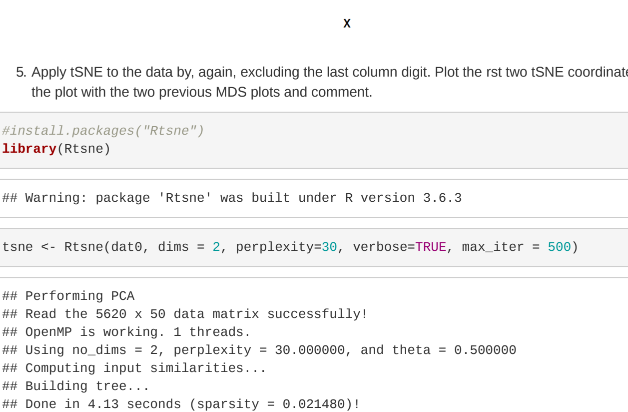
Patters are not observed. It looks cloud of numbers here and there.

4. Try out another MDS method of your own choice, e.g., Sammon mapping, non-metric MDS. Note that the last column containing the target variable digits should be excluded from the analysis. Plot the rst two low-dimensional coordinates. Again, highlight di erent digits with both color and symbols.

```
# euclidean distances between the rows
d = dist(dat0)
# k is the number of dim
library("MASS")
fit = isoMDS(d, k=2)
```

```
## initial  value 33.723183
## final  value 33.722680
## converged
```

```
#fit # view results
# plot solution
x = fit$points[,1]; y <- fit$points[,2]
 plot(x,y,pch=as.character(dat$digit),col=colr,main= "Nonmetric MDS")
```



5. Apply tSNE to the data by, again, excluding the last column digit. Plot the rst two tSNE coordinates by highlighting di erent digits. Compare the plot with the two previous MDS plots and comment.

```
#install.packages("Rtsne")
library(Rtsne)
```

```
## Warning: package 'Rtsne' was built under R version 3.6.3
```

```
tsne <- Rtsne(dat0, dims = 2, perplexity=30, verbose=TRUE, max_iter = 500)
```

```
## Performing PCA
## Read the 5620 x 50 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 30.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 4.13 seconds (sparsity = 0.021480)!
## Learning embedding...
## Iteration 50: error is 90.736096 (50 iterations in 1.17 seconds)
## Iteration 100: error is 74.474810 (50 iterations in 1.16 seconds)
## Iteration 150: error is 71.143984 (50 iterations in 0.96 seconds)
## Iteration 200: error is 70.102962 (50 iterations in 1.05 seconds)
## Iteration 250: error is 69.594957 (50 iterations in 1.13 seconds)
## Iteration 300: error is 2.331297 (50 iterations in 0.96 seconds)
## Iteration 350: error is 1.962640 (50 iterations in 0.89 seconds)
## Iteration 400: error is 1.763780 (50 iterations in 0.98 seconds)
## Iteration 450: error is 1.641694 (50 iterations in 0.87 seconds)
## Iteration 500: error is 1.558808 (50 iterations in 0.91 seconds)
## Fitting performed in 10.08 seconds.
```

```
plot(tsne$Y[,1], tsne$Y[,2], main="tsne",col=colr,pch=as.character(dat$digit),
     xlab = "comp1",ylab = "comp2")
```