Dimensionality Reduction and Visualization
Exercise set 8 Solutions.

Md. Abdullah-Al Mamun

Problem E6: Self-Organizing Map

Part 1: Self Organizing Map vs Vector Quantization.

Self-Organizing Map vs Vector Quantization are following:

Self-organizing Map (SOM) = $u(t+1) = u(t) + hu, v(t). a(t).x(t) – u(t)$

$h_{u,v}(t) = \exp(-\frac{d_{grid}}{\sigma(t)})$ and $\sigma(t) = \sigma_0 \exp(-t/\lambda)$

Vector Quantization: $v(x(t)) = v(x(t)) + a(t).(x(t) – v(x(t))$

Where, t denotes infinity, σ(t) tends to zero

$h_{u,v}(t)$ denotes 0 when $u \neq v$ and 1

u = v when $d_{grid}(u,v) = 0$.

So, the iteration 0f SOM only works on the nearest prototype v in the same way of vector quantization.

Part 2: Self-organizing Map of Kidney disease patients

Plot how the “blood pressure” and “red blood cell count” attributes vary over the prototype vectors of the resulting Self-organizing map.

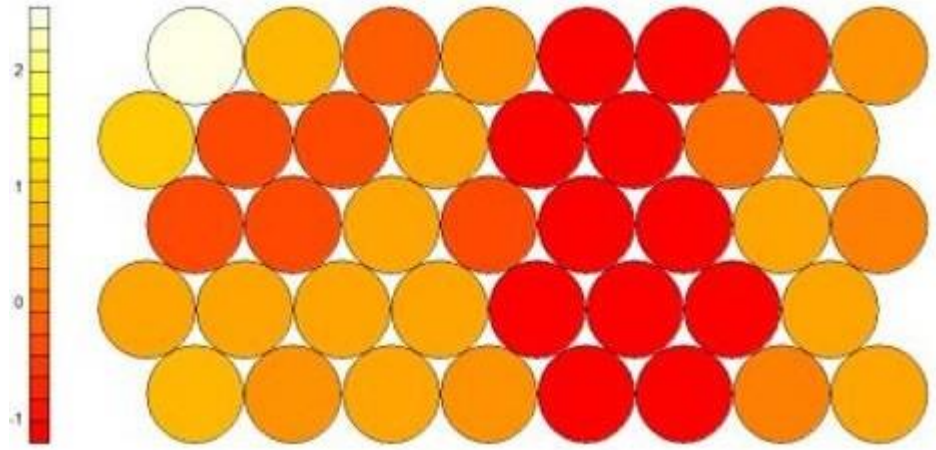


Figure 6: blood pressure (BP)

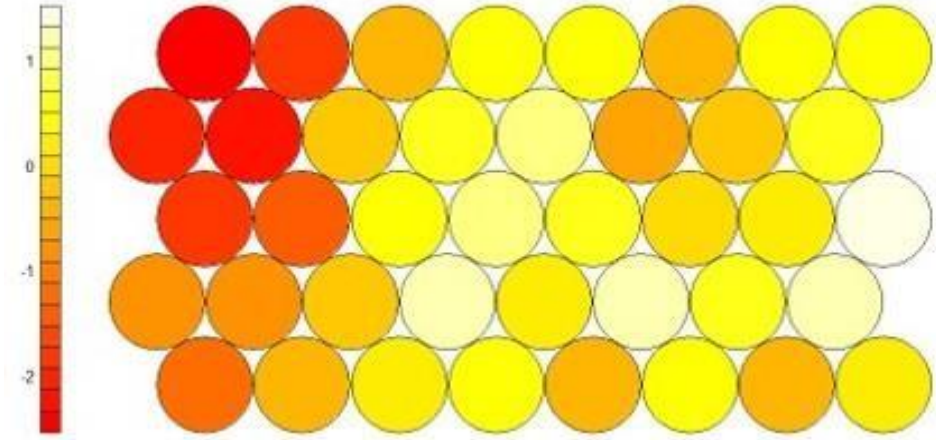


Figure 7: red blood cell (RBC) count

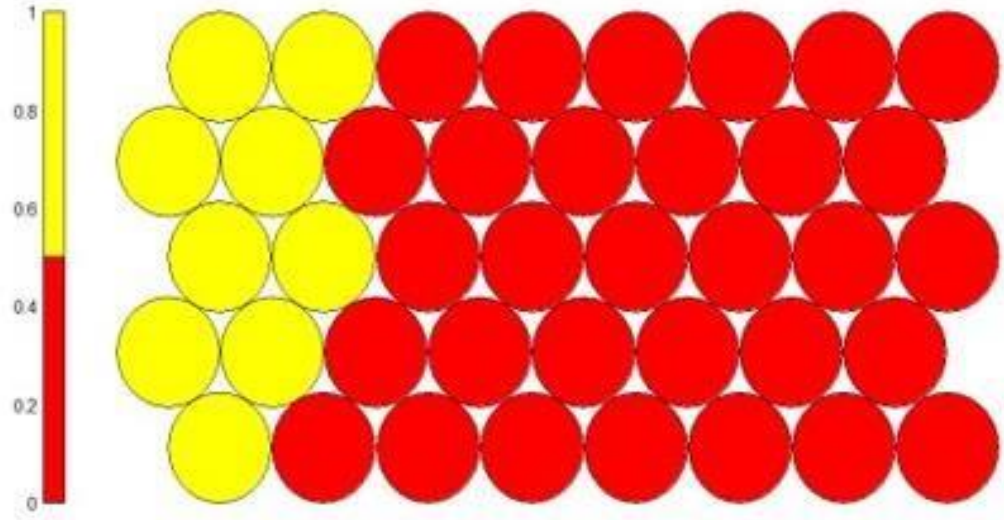
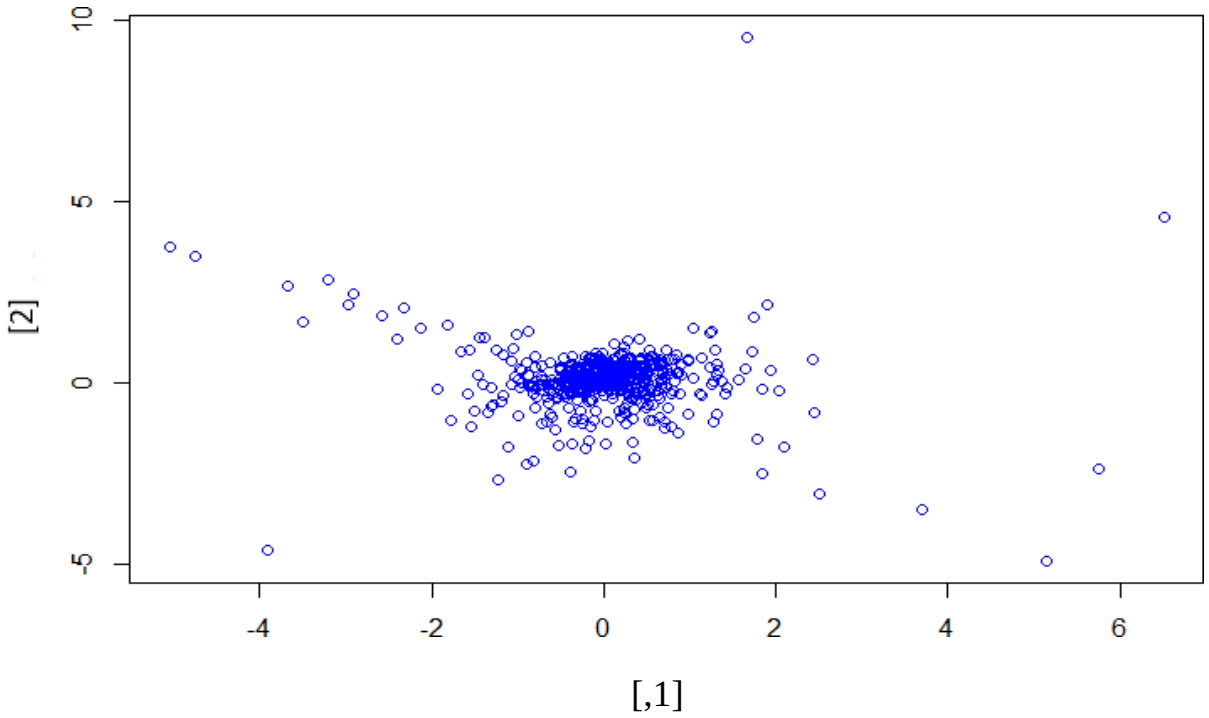
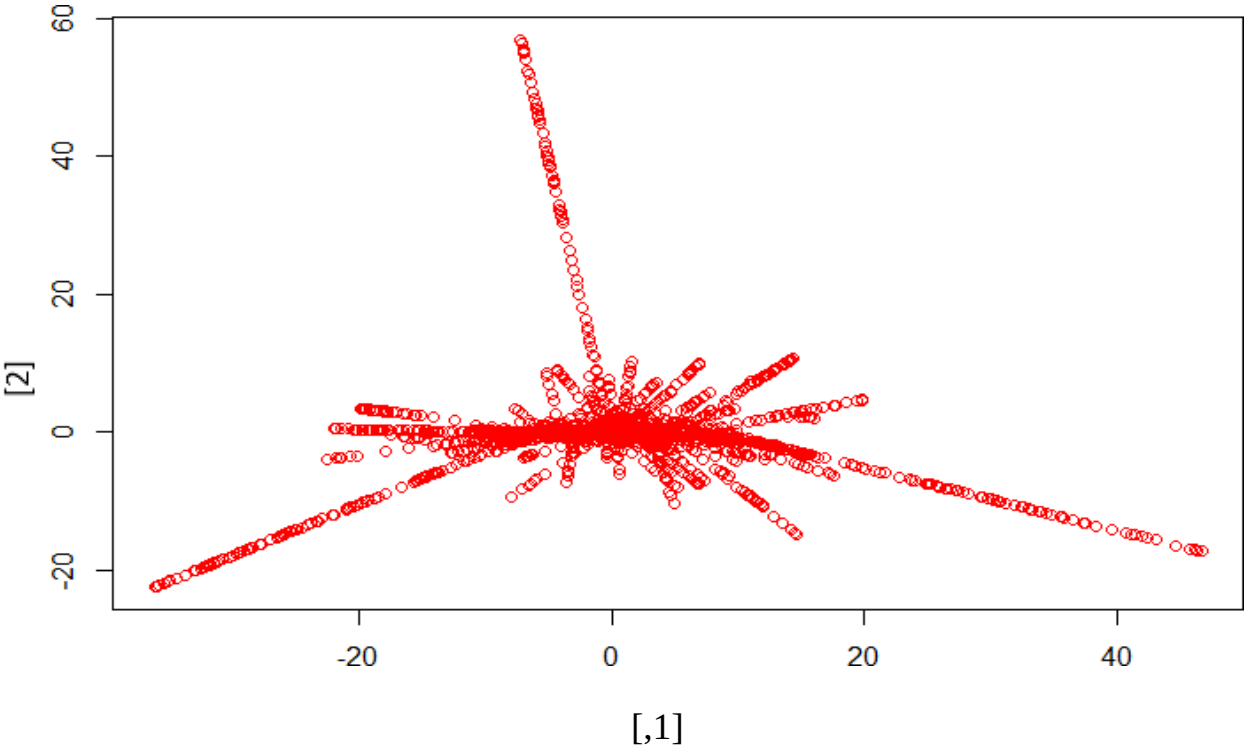


Figure 8:The plot of majority class of data at each prototype step (“ckd” or “notckd”)

Problem E7: Laplacian Eigenmap, Isomap, Locally Linear Embedding, and and Precision and Recall.

Part 1:





Index

In [ ]:

```
data <- read.delim(file.choose(), header = TRUE, sep="", skip=2, as.is=TRUE)
test=read.arff(file.choose(), header = TRUE, sep="", skip=2, as.is=TRUE)
install.packages("RWeka")
install.packages("R.matlab")
install.packages("BiocManager")
BiocManager::install("Rgraphviz")

source("http://bioconductor.org/biocLite.R")
biocLite("RDRToolbox")
library(kohonen)
library(RWeka)
library(R.matlab)
library(RDRToolbox)
library(Rgraphviz)
library(foreign)
```

E6: Self-Organizing Map

Part 2: Self-organizing map of kidney disease patients.

In [ ]:

```
data0=read.csv("Chronic_Kidney_Disease/chronic_kidney_disease_full.csv",stringsA
sFactors=FALSE) # "WEKA"package was used to import "arff" file into "csv" file.
data1=data0[146:400,] # removing the first 145 rows as stated in this question
ind=c()
for (i in 1:255){
  v=sum(data1[i,]=="?")
  if (v==0){
    ind=append(ind,i)
  }
}
data2=data1[ind,]
indCol = which(is.na(as.numeric(as.character(data2[1,])))) # finding the column
 of char type
for (i in indCol){
  data2[,i]=as.character(as.numeric(as.factor(data2[,i])))
}
data3=(as.matrix(data2)) # Converting from dataframe to matrix
data4=apply(data3,2,as.numeric) # Converting from char matrix to numeric matrix
data5=data4
data5[,1:24] = scale(data4[,1:24]) # normalize each column with mean zero and sd
=1

# Creating a training data set
data_train <- data5[, c(1:2,10:18)] #Only use the 11 numerical attributes

# Creating the SOM Grid
som_grid <- somgrid(xdim = 8, ydim=5, topo="hexagonal")

# Training the SOM, and options for the number of iterations,
# the learning rates, and the neighbourhood are available
som_model <- som(data_train, grid=som_grid, rlen=1000, alpha=c(0.05,0.01),
                 keep.data = TRUE,n.hood="circle")

# Plotting how the blood pressure and red blood cell count attributes vary

# over the prototype vectors of the resulting SOM.
plot(som_model, type = "property",
     property = som_model$codes[,2], main="blood pressure")
plot(som_model, type = "property",
     property = som_model$codes[,11], main="red blood cell count")

# Plotting also what is the majority class of data at each prototype vector (ckd
or notckd").
som.prediction <- predict(som_model, newdata = data_train,
                          trainX = data_train,
                          trainY = factor(data5[,25]))

str(som.prediction)
isCKD=  (som.prediction$unit.predictions[,1]>=0.5)+0

# Majority class of data at each prototype vector: 1 for ckd and 0 for nockd.
plot(som_model,type = "property",
     property = isCKD,main="Red:notckd & Yellow:ckd")
```

E7

In [ ]:

```
library(RDRToolbox)
swiss_data = SwissRoll(N=2000,Height = 30, Plot=FALSE)
#is.matrix(swiss_data)
dim(swiss_data)
mj = LLE(swiss_data, dim=2, k=2)
plot(mj, col="blue")

simData_dim2_IM = Isomap(data=swiss_data, dims=2, k=2, col="blue")
plot(simData_dim2_IM$dim2)
simData_dim1to10_IM = Isomap(data=swiss_data, dims=1:10, k=2, plotResiduals=TRUE
, col="blue")
plot(simData_dim1to10_IM$dim2)
```

```
nePCA$x[,3:4], col= classes) # Plotting PCA3 and PCA4 only; 3 colors
```