Week 6 Code Examples, CSC 400, Spring 2024

1. Hierarchical Clustering
2. Agglomerative Clustering
3. Spectral Clustering
4. Visualization

Hierarchical Clustering

We'll start with a short example of hierarchical clustering on the mtcars dataset. Clustering is normally done on data with no labels, so unlike classification, we have nothing to match. We'll use the Euclidean distance here, but this is not the only option. Another step to consider is rescaling the data.

```
library(dplyr)
head(mtcars)

distance_mat <- dist(mtcars, method = 'euclidean')
distance_mat

set.seed(240)
Hierar_cl <- hclust(distance_mat, method = "average")
Hierar_cl

plot(Hierar_cl)
abline(h = 110, col = "green")
fit <- cutree(Hierar_cl, k = 3 )
fit
table(fit)
rect.hclust(Hierar_cl, k = 3, border = "green")
```

Agglomerative Clustering

We load the data and split off X and Y (response). Since we are clustering, we won't use Y until we are comparing in a semi-supervised process.

```
data(pima)
RawData<-pima
responseY = RawData[,dim(RawData)[2]]
predictorX = RawData[,1:(dim(RawData)[2]-1)]
```

We build our model and plot the resulting dendrogram.

```
library(cluster)
agn = agnes(x=predictorX[1:25,], diss = FALSE, stand = TRUE,
            method = "average")
DendAgn =as.dendrogram(agn)
plot(DendAgn)
```

We can change the way that the clusters are collected.

```
agn = agnes(x=predictorX[1:25,], diss = FALSE, stand = TRUE,
            method = "single")
DendAgn =as.dendrogram(agn)
plot(DendAgn)
```

```
agn = agnes(x=predictorX[1:25,], diss = FALSE, stand = TRUE,
            method = "complete")
DendAgn =as.dendrogram(agn)
plot(DendAgn)
```

Spectral Clustering

Let's look at some example data.

```
library(mlbench)

set.seed(111)
obj <- mlbench.spirals(100,1,0.025)
my.data <-  4 * obj$x
plot(my.data)
```

We then build the similarity matrix.

```
s <- function(x1, x2, alpha=1) {
  exp(- alpha * norm(as.matrix(x1-x2), type="F"))
}

make.similarity <- function(my.data, similarity) {
  N <- nrow(my.data)
  S <- matrix(rep(NA,N^2), ncol=N)
  for(i in 1:N) {
    for(j in 1:N) {
      S[i,j] <- similarity(my.data[i,], my.data[j,])
    }
  }
  S
}

S <- make.similarity(my.data, s)
S[1:8,1:8]
```

And then we build the affinity matrix.

```
make.affinity <- function(S, n.neighboors=2) {
  N <- length(S[,1])

  if (n.neighboors >= N) {  # fully connected
    A <- S
  } else {
    A <- matrix(rep(0,N^2), ncol=N)
    for(i in 1:N) { # for each line
      # only connect to those points with larger similarity
      best.similarities <- sort(S[i,], decreasing=TRUE)[1:n.neighboors]
      for (s in best.similarities) {
        j <- which(S[i,] == s)
        A[i,j] <- S[i,j]
        A[j,i] <- S[i,j] # to make an undirected graph, ie, the matrix becomes symmetric
      }
    }
  }
  A
}

A <- make.affinity(S, 3)  # use 3 neighboors (includes self)
A[1:8,1:8]
```

Then we need to complete the math for the process.

```
D <- diag(apply(A, 1, sum)) # sum rows
D[1:8,1:8]
U <- D - A
round(U[1:12,1:12],1)

"%^%" <- function(M, power)
  with(eigen(M), vectors %*% (values^power * solve(vectors)))

k   <- 2
evL <- eigen(U, symmetric=TRUE)
Z   <- evL$vectors[,(ncol(evL$vectors)-k+1):ncol(evL$vectors)]
plot(Z, col=obj$classes, pch=20)
```

The plot produces two clusters, but let's visualize this in the context of the original data.

```
library(stats)
km <- kmeans(Z, centers=k, nstart=5)
plot(my.data, col=km$cluster)

signif(evL$values,2)
plot(1:10, rev(evL$values)[1:10], log="y")
abline(v=2.25, col="red", lty=2)
```

The package kernlab has all this programming in one built-in function.

```r
library(kernlab)

sc <- specc(my.data, centers=2)
plot(my.data, col=sc, pch=4)              # estimated classes (x)
points(my.data, col=obj$classes, pch=5) # true classes (<>)
```

Resources:

1. https://www.r-bloggers.com/2016/01/hierarchical-clustering-in-r-2/
2. https://uc-r.github.io/hc_clustering
3. https://www.di.fc.ul.pt/~jpn/r/spectralclustering/spectralclustering.html