Prof. Dr. Steffen Wagner
Angewandte Statistik
Fachbereich II
Berliner Hochschule für Technik

Exercise 03: Hierarchical and Soft Clustering

Machine Learning I – SoSe 24

This workshop covers hierarchical clustering and soft clustering. At the end of the worksheet there are a couple of written exercises for you to do at home, which should be good practice for the exam.

1 Preparations

1.1 RStudio Project

1. Open your Machine Learning 1 RStudio Project
2. Create an R Script file to perform this exercise

1.2 Required Packages

For this exercise you require the following additonal R packages. Please make sure that you have installed them on your computer before coming to the workshop session since Eduroam is still not working.

```
# check if packages can be loaded, i.e. they are already installed
library(EMCluster)       # Soft CLustering
library(mvtnorm)         # multi-variate normal distributions
library(ISLR2)           # contains the `NCI60` data set
```

If you get an error at this stage, you need to install the packages.

2 Hierarchical Clustering

2.1 Introduction

This exercise is a short introduction to hierarchical clustering. First recreate the artificial data set using.

```
# data generation
set.seed(1234567890)
x <- matrix(rnorm(50 * 2), ncol = 2)
x[1:25,1] <- x[1:25,1] + 2
x[1:25,2] <- x[1:25,2] - 2
```

Then work through pages 538/539 for an introduction to *Hierarchical Clustering* in *James et al*.

2.2 The `NCI60` data set

This is an unsupervised learning analysis of a fairly complicated genomic data set is using PCA and hierarchical clustering. It is covered in *James et al.* on pages 540 to 545[1].

[1]You can download the code in the labs from https://web.stanford.edu/~hastie/ISLR2/Labs/R_Labs/Ch12-unsup-lab.R. Of course you should still read the accompanying text in the book when working through the code!

3 Soft CLustering

3.1 EM Clustering by hand

The purpose of this exercise is for you to investigate how the EM soft clustering works, in particular the E and M steps. The code is provided for you in the file `EM_Clustering_by_hand.R` in Moodle[2] and is well commented. You should read the comments that describe what the code does. In some places you will need to replace `???` with the correct code.

a) You will generate two clusters that overlap. The K-Means approach gets some values wrong.
b) The initialisation can make a difference, so I have chosen two centroids one in the left half and one in the right half of the x-axis, both centroids have the same initial y-axis value. The initial variance matrix assumes the same variance in both clusters and no correlation.
c) Once centroids and variances are defined, the probabilities are calculated using the function `dmvnorm`, which is in the package `mvtnorm`.
d) Repeat these iterative steps by hand a few times.
e) Update the code to include a for loop with 25 iterations.

3.2 Soft Clustering using `EMCluster`

Apply the functionality of the package `EMCluster` to the example data constructed in exercise 3.1. The obtained clusters differ using `EMCluster` compared to the above code. The results appear to be better than the k-means results.

3.3 EM Clustering using the `USArrests` data

This is a short exercise for you to learn more about the Soft clustering in R using the EM-Algortihm.
You will use the `USArrests` data, because you are familiar with the data and you can easily compare the K-Means results with the EM-Clustering results.

a) First of all create an object with the scaled `USArrests` data, as you will be using it multiple times.

b) Then Run a K-means clustering with 4 clusters and plot the clusters for the first 2 principal components.

```r
data("USArrests")
USArr_scaled <- scale(USArrests)
km_out <- kmeans(USArr_scaled, centers = 4, nstart = 20)
pca_out <- prcomp(USArr_scaled)
plot(pca_out$x[, 1:2], type = "n")
text(pca_out$x[, 1:2], labels = state.abb, col = km_out$cluster)
```

c) The approach in the `EMCluster` package is to set up an initial solution using the function `init.EM` and then run the EM algorithm to get the optimal solution using the `EMCluster` function. In the third step we obtain the probabilities of the optimal solution by running once more `e.step`.

```r
library(EMCluster)
em_obj<- init.EM(USArrests, nclass = 4)
```

[2] https://lms.bht-berlin.de/mod/resource/view.php?id=1191015

```r
em_cl_obj <- emcluster(USArrests, em_obj, assign.class = TRUE)
em_cl_probs <- e.step(USArrests, emobj = em_cl_obj)$Gamma
```

d) Visualisiation: plot the 1st 2 principal components again and compare this with the K-Means plot.

```r
plot(pca_out$x[, 1:2], type = "n")
text(pca_out$x[, 1:2], labels = state.abb, col= em_cl_obj$class)
```

Another perspective is looking at the 2-dim contingency table:

```r
addmargins(
  table(em_cl_obj$class, km_out$cluster, dnn = c("EM", "M-Means"))
)
```

e) We can look at each K-means cluster in more detail by returning the EM-algorithm probabilities for each cluster, and the mean values for each cluster. Here is the code for the 1st cluster:

```r
round(em_cl_probs[km_out$cluster == 1, ], 2)
round(apply(em_cl_probs[km_out$cluster == 1, ], 2, mean), 2)
```

Repeat this for each of the four K-means clusters. Notice that the two methods agree well for some but not all clusters.

4 Written Exercises

4.1 Hierarchical Clustering

Below are 7 data points in 2-dimensions.

Element	X_1	X_2
1	5	1
2	8	-2
3	8	1
4	9	7
5	13	-2
6	17	3
7	18	5

a) Calculate the Euclidean distances between each pair of points rounded to 1 decimal place. You can use the R function `dist()` to do this.

```
##      1    2    3    4    5    6
## 2  4.2
## 3  3.0  3.0
## 4  7.2  9.1  6.1
## 5  8.5  5.0  5.8  9.8
## 6 12.2 10.3  9.2  8.9  6.4
## 7 13.6 12.2 10.8  9.2  8.6  2.2
```

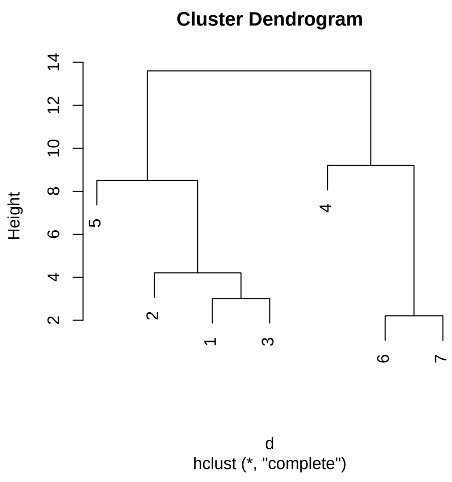
b) Use the distance matrix to find the first cluster join. Write down the two element numbers and the distance between them.

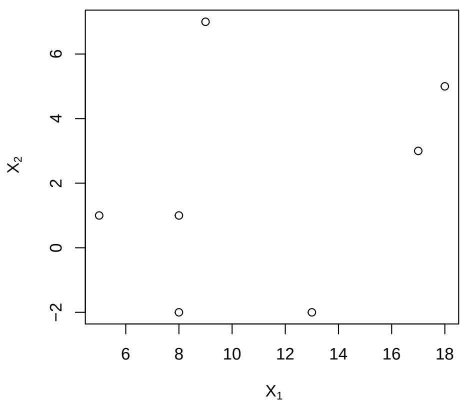
c) Using the complete linkage rule repeat step (b) to obtain the hierarchical clusters. At each step write down the element numbers for each element in the new cluster and the distance used to determine the cluster.

d) Draw the scatter plot of the points and the dendrogram of the clustering.

Solution:

Elements	Distance
6-7	2.2
1-3	3.0
1-2-3	4.2
1-2-3-5	8.5
4-6-7	9.2
1-2-3-4-5-6-7	13.6


Cluster Dendrogram
Height
2
4
6
8
10
12
14
5
2
1
3
4
6
7
d
hclust (*, "complete")


X_2
X_1
−2
0
2
4
6
6
8
10
12
14
16
18

4.2 Example Exam Question: K-Means and K-Medoids

This exercise was in part of a exam in a previous year.

a) What is the principal difference between the K-Means and the K-Medoids clustering algorithms?

b) What advantage is gained when using K-Medoids instead of K-Means?

c) Why is it undesirable to adapt the K-Means algorithm by replacing the cluster means with cluster medians?

d) Give an outline of the K-Medoids (PAM) algorithm.

e) The x and y coordinates of a data set with 8 points in $\mathbb{R}^2$ is given in the following table.

x	y	initial cluster
3	8	
5	2	
3	4	
7	6	
8	8	
9	5	
10	9	
10	7	

The *squared distances* between each pair of points is given is as follows:

```
##     1   2   3   4   5   6   7
## 2  40
## 3  16   8
## 4  20  20  20
## 5  25  45  41   5
## 6  45  25  37   5  10
## 7  50  74  74  18   5  17
## 8  50  50  58  10   5   5   4
```

Points 3 and 8 have been chosen at random as the initial $K{=}2$ medoids.

- Assign the initial clusters. Enter your answers in the table above.
- Calculate the total within medoid distance using squared Euclidean distance as the distance metric.
- Without further calculation explain the next iteration in the PAM algorithm using this data set.

Solution:

a) Instead of finding the Centroid of each cluster, a number of points are assigned as Medoids, and the clusters are defined as the points whose distance is closest to the corresponding medoid.

b) The algorithm is more robust with respect to outliers.

c) Because the median will have to be repeatedly calculated. The median requires sorting the distances which is computationally intensive.

d) Algorithm:

i) Choose k data points (at random) to be the initial k medoids.
ii) Assign each point to its nearest medoid, the one with the smallest *distance*. This defines the current clusters. Store the total within medoid distance W.
iii) Choose one data point to be a potential new medoid and repeat step 2.
  - If this new medoid gives a decreased W, accept this as the new medoid, drop the old medoid that was in the same cluster.
  - If this new medoid does not decrease W reject the proposal and move to another point.

iv) Repeat iii) until no better medoid can be found.

e) The initial clusters are as follows:

x	y	initial cluster
3	8	1
5	2	1
3	4	1
7	6	2
8	8	2
9	5	2
10	9	2
10	7	2

and the total within medoid distance results to

```
## Cluster 1 Cluster 2
##        56       214
```

Next iteration step: see steps iii) and iv) above.