Clustering methods: Hierarchical

PGDM Research and Analytics cell
Madras School of Economics
Reference - An introduction to Statistical Learning

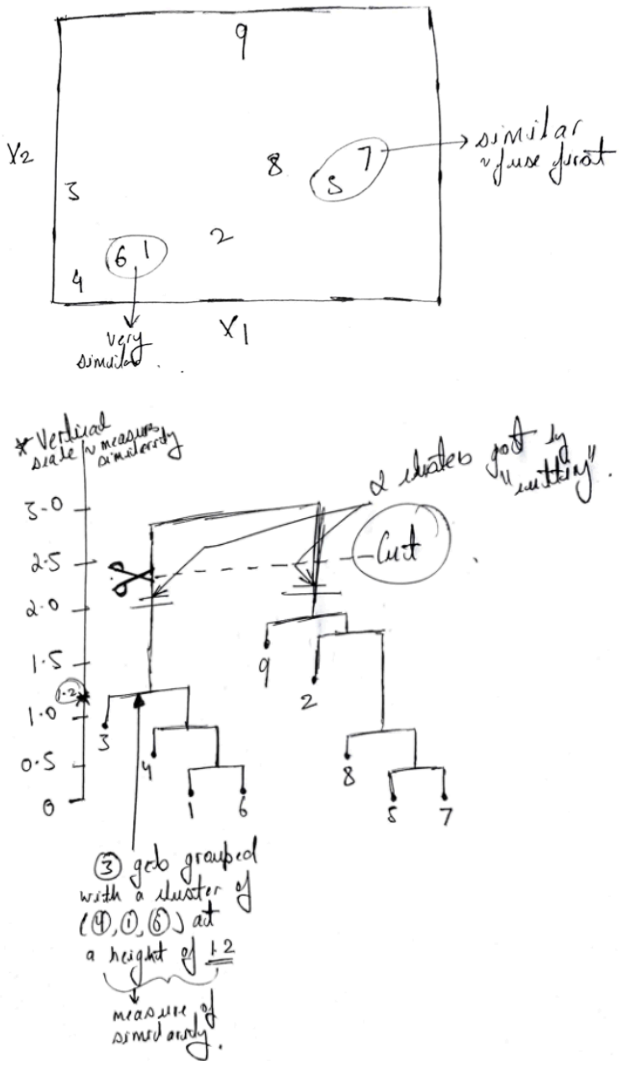
Hierarchical Clustering

In this approach we are not required to pre-specify a number of clusters to partition our dataset into groups. It is a **bottom up (agglomerative)** clustering method that results in a graphical representation of clusters called the **Dendrogram**. This dendrogram can be thought of as an **inverted tree** where we keep combining leaves and then ultimately reach the trunk (one node/cluster).

Interpreting a Dendrogram

Suppose we have 45 observations in our data that we want to cluster. Each of the 45 observations is initially thought of as a **leaf**. As we move up the inverted tree, these **leaves** get **fused** into branches. Observations that get fused are considered **similar to each other**. And as we move further up, the branches also fuse with other branches. Note some important points:

1. observations that fuse **later** (more upward) in the inverted tree - tend to be more and more differet as compared to observations or branches that fuse **earlier** in the inverted tree.
2. We can look at the points along the inverted tree where a pair of observations or branches get fused - the height of this point on the vertical axis, tells us how similar or different the observations are.
3. To obtain clusters - we **cut** the dendrogram at a desired level - and all the set of observations beneath this cut represent our clusters. The appropriate descriptive figures are given below:



The alorithm

1. Before starting out, we measure the **dissimilarity** between observations using **Euclidean distance**.
2. Start at the bottom of the dendrogram where each observation **is its own cluster**. So for n observations we initially have n clusters.
3. Next, the two closest observations are **fused** into one cluster - hence we now have $n - 1$ clusters. Then another pair of clusters/observations are fused - resulting in $n - 2$ clusters. We iteratively proceed in this fashion to reduce ultimately 1 cluster - which is the trunk of the dendrogram - this completes our inverted tree.
4. When we try to fuse observations with each other its simply - use smallest euclidean distance. But when we try to fuse a cluster, say $\{x, y\}$ with another observation $\{z\}$ then we use a measure of **dissimilarity** known as **linkage**.
5. In **Complete linkage** we compute pairwise euclidean similarities between observations of

cluster A and cluster B and record the **largest** of these distances as the **distance between cluster A and cluster B**.

6. In **Single linkage** we compute pairwise euclidean similarities between observations of cluster A and cluster B and record the **smallest** of these distances as the **distance between cluster A and cluster B**.
7. In **Average linkage** we compute pairwise euclidean similarities between observations of cluster A and cluster B and record the **average** of these distances as the **distance between cluster A and cluster B**.
8. In **Centroid linkage** we compute the centroid of observations in A and the centroid of observations in B - the distance between the **two centroids is the distance between cluster A and cluster B**.

Simulation

We use the same data that was used in the previous note on Kmeans - 50 random observations sampled from a normal distribution with the first 25 observations being shifted.

```
set.seed(2)
x = matrix(rnorm(50*2), ncol = 2)

### shifting the mean (+3) of the 1st 25 obs in 1st col
x[1:25, 1] = x[1:25, 1] + 3

### shiting the mean (-4) of the 2nd 25 obs in 2nd col
x[1:25, 2] = x[1:25, 2] - 4

# taking a look at the data
x
```

```
##              [,1]         [,2]
##  [1,]  2.10308545 -4.838287148
##  [2,]  3.18484918 -1.933698644
##  [3,]  4.58784533 -4.562247053
##  [4,]  1.86962433 -2.724284488
##  [5,]  2.91974824 -5.047572627
##  [6,]  3.13242028 -5.965878241
##  [7,]  3.70795473 -4.322971094
##  [8,]  2.76030198 -3.064137473
##  [9,]  4.98447394 -2.860770197
## [10,]  2.86121299 -2.328381233
## [11,]  3.41765075 -5.788242207
## [12,]  3.98175278 -1.968757481
## [13,]  2.60730464 -4.703144333
```

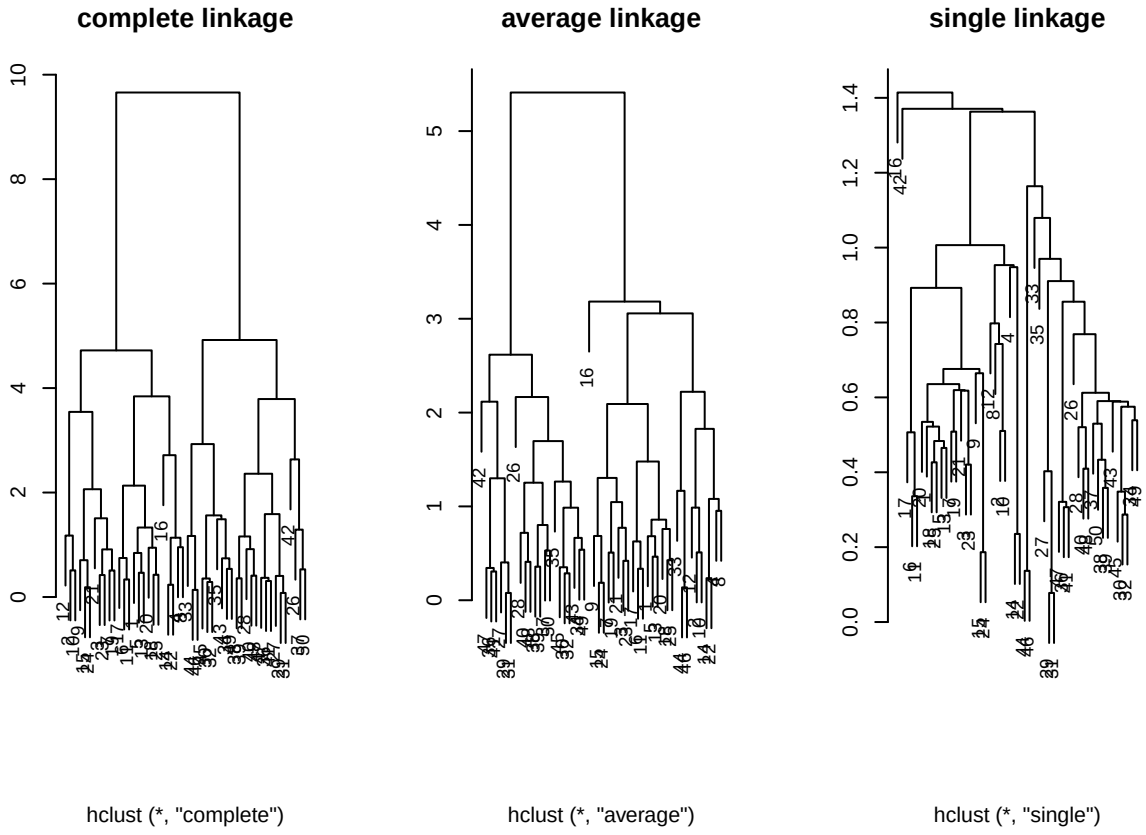
```
## [14,]  1.96033102 -3.841835237
## [15,]  4.78222896 -3.493765203
## [16,]  0.68893092 -4.819995106
## [17,]  3.87860458 -5.998846995
## [18,]  3.03580672 -4.479292591
## [19,]  4.01282869 -3.915820096
## [20,]  3.43226515 -4.895486611
## [21,]  5.09081921 -4.921275666
## [22,]  1.80007418 -3.669550497
## [23,]  4.58963820 -4.141660809
## [24,]  4.95465164 -3.565152238
## [25,]  3.00493778 -4.053722626
## [26,] -2.45170639 -0.907110376
## [27,]  0.47723730  1.303512232
## [28,] -0.59655817  0.771789776
## [29,]  0.79220327  1.052525595
## [30,]  0.28963671 -1.410038341
## [31,]  0.73893860  0.995984590
## [32,]  0.31896040 -1.695764903
## [33,]  1.07616435 -0.533372143
## [34,] -0.28415772 -1.372269451
## [35,] -0.77667527 -2.207919779
## [36,] -0.59566050  1.822122519
## [37,] -1.72597978 -0.653393411
## [38,] -0.90258448 -0.284681219
## [39,] -0.55906191 -0.386949604
## [40,] -0.24651257  0.386694975
## [41,] -0.38358623  1.600390852
## [42,] -1.95910318  1.681154956
## [43,] -0.84170506 -1.183606388
## [44,]  1.90354747 -1.358457254
## [45,]  0.62249393 -1.512670795
## [46,]  1.99092044 -1.253104899
## [47,] -0.30548372  1.959357077
## [48,] -0.09084424  0.007645872
## [49,] -0.18416145 -0.842615198
## [50,] -1.19876777 -0.601160105
```

We employ the **hclust** function to carry out hierarchical clustering and the **dist** function to compute the 50×50 matrix of pairwise euclidean distances among the 50 observations in our data. The code is demonstrated to use three different linkages for comparison.

```
hc.complete = hclust(dist(x), method="complete")
hc.average = hclust(dist(x), method="average")
hc.single = hclust(dist(x), method="single")
```

```r
# plotting the dendrograms for all three methods

par(mfrow=c(1, 3))
plot(hc.complete, main="complete linkage", xlab="", ylab="", cex=0.9)
plot(hc.average, main="average linkage", xlab="", ylab="", cex=0.9)
plot(hc.single, main="single linkage", xlab="", ylab="", cex=0.9)
```



Now in order to find the clusters and their associated numbers, we **cut** the tree using the **cutree** function.

```r
# cutting at a height of 2 - obtain 2 clusters
cutree(hc.complete, 2)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2
```

```r
cutree(hc.average, 2)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 1 2 2 2 2
## [39] 2 2 2 2 2 1 2 1 2 2 2 2
```

```r
cutree(hc.single, 2)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [39] 1 1 1 1 1 1 1 1 1 1 1 1
```

```
## [39] 1 1 1 1 1 1 1 1 1 1 1 1
```

NOTE that in the **single linkage** tree, we are getting almost all 1 labels - this is because a **cut at the height of** 2 is not enought to form two clear clusters in this case. So we try by cutting the **single linkage tree at a height of** 4 to get distinct clusters.

```
cutree(hc.single, 4)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3
## [39] 3 3 3 4 3 3 3 3 3 3 3 3
```