TMA4268 Statistical Learning

Chapter 10: Unsupervised Learning

Thiago G. Martins, Department of Mathematical Sciences, NTNU

Spring 2022

Contents

Lab 2: Clustering

K-Means Clustering

Simulate data

Lets simulate data with the intention to create two clusters:

```r
set.seed(2)
x=matrix(rnorm(50*2), ncol=2)
x[1:25,1]=x[1:25,1]+3
x[1:25,2]=x[1:25,2]-4
```

Perform k-means

The function `kmeans()` performs K-means clustering in R. We now perform K-means clustering with K = 2.

```r
km.out=kmeans(x,2,nstart=20)
```

To run the `kmeans()` function in R with multiple initial cluster assignments, we use the `nstart` argument. The `kmeans()` function will report only the best results.

Cluster assignments

The cluster assignments of the 50 observations are contained in `km.out$cluster`:

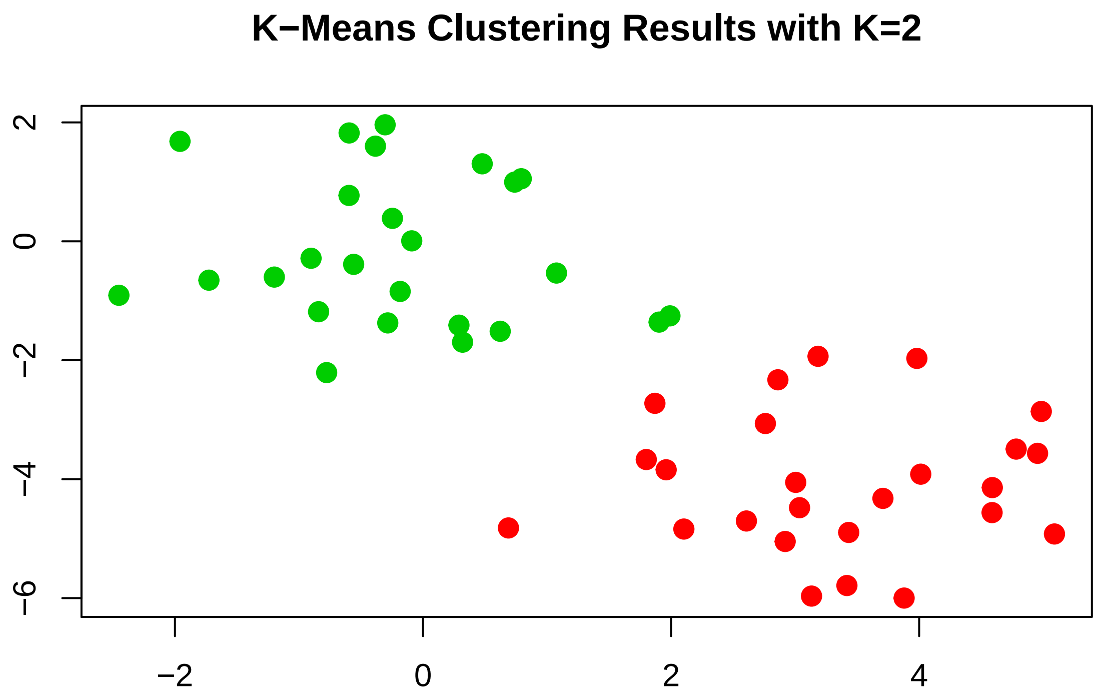
```r
km.out$cluster
```

```
##  [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 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Plot the data

We can plot the data, with each observation colored according to its cluster assignment.

```r
plot(x,
     col=(km.out$cluster+1),
     main="K-Means Clustering Results with K=2",
     xlab="", ylab="", pch=20, cex=2)
```



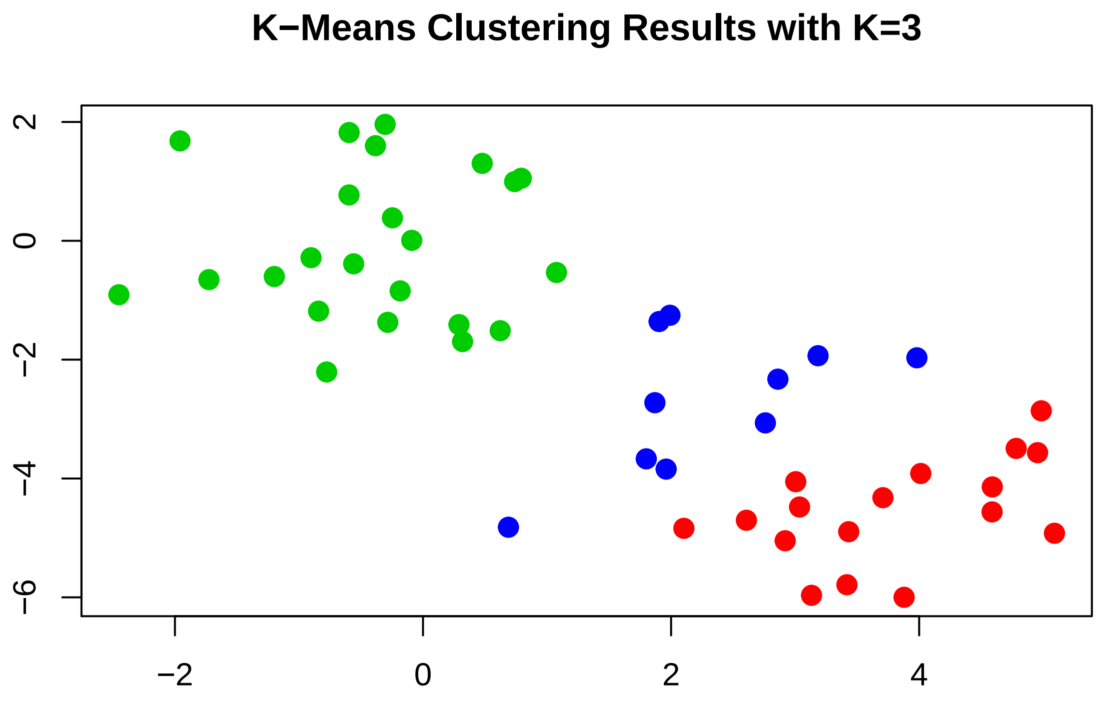
kmeans with k = 3

However, for real data, in general we do not know the true number of clusters. We could instead have performed K-means clustering on this example with K = 3.

```
set.seed(4)
km.out=kmeans(x,3,nstart=20)
km.out
```

```
## K-means clustering with 3 clusters of sizes 17, 23, 10
##
## Cluster means:
##         [,1]        [,2]
## 1  3.7789567 -4.56200798
## 2 -0.3820397 -0.08740753
## 3  2.3001545 -2.69622023
##
## Clustering vector:
##  [1] 1 3 1 3 1 1 1 3 1 3 1 3 1 3 1 3 1 1 1 1 1 3 1 1 1 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 3 2 3 2 2 2 2
##
## Within cluster sum of squares by cluster:
## [1] 25.74089 52.67700 19.56137
##  (between_SS / total_SS =  79.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```
plot(x, col=(km.out$cluster+1), main="K-Means Clustering Results with K=3", xlab="", ylab="", pch=20, c
```



Multiple starting points

We compare using nstart=1 to nstart=20.

```
set.seed(3)
km.out=kmeans(x,3,nstart=1)
km.out$tot.withinss
```

```
## [1] 97.97927
```

```
km.out=kmeans(x,3,nstart=20)
km.out$tot.withinss
```

```
## [1] 97.97927
```

Note that `km.out$tot.withinss` is the total within-cluster sum of squares, which we seek to minimize by performing K-means clustering.

We strongly recommend always running K-means clustering with a large value of nstart, such as 20 or 50, since otherwise an undesirable local optimum may be obtained.

Hierarchical Clustering

The `hclust()` function implements hierarchical clustering in R. We will use the data simulated in the K-means section.

Next, we will compute hierarchical clustering dendrogram using complete, single, and average linkage clustering, with Euclidean distance as the dissimilarity measure.

Perform hierarchical clustering

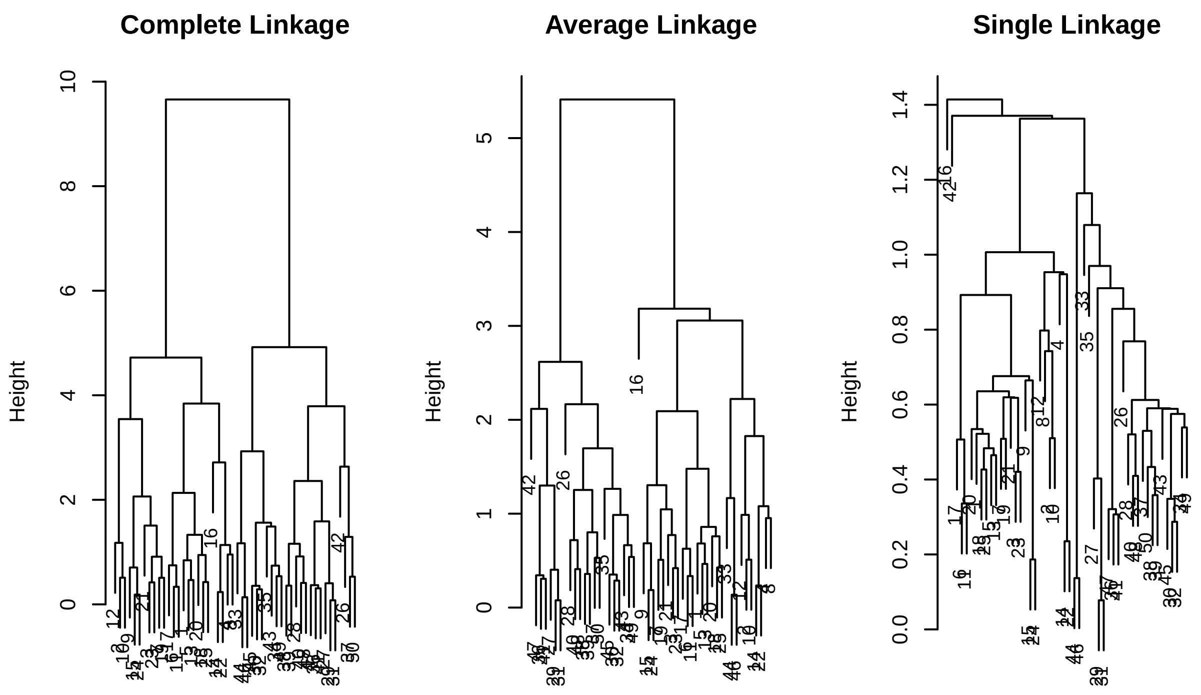
The `dist()` function is used to compute the 50×50 inter-observation Euclidean distance matrix.

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

Plot dendograms

We can now plot the dendrograms obtained using the usual `plot()` function. The numbers at the bottom of the plot identify each observation.

```
par(mfrow=c(1,3))
plot(hc.complete,main="Complete Linkage", xlab="", sub="", cex=.9)
plot(hc.average, main="Average Linkage", xlab="", sub="", cex=.9)
plot(hc.single, main="Single Linkage", xlab="", sub="", cex=.9)
```



Cluster assignment

To determine the cluster labels for each observation associated with a given cut of the dendrogram, we can use the `cutree()` function:

```
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 1 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
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 2 1 2 2
## [36] 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2
```

```
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
## [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

For this data, complete and average linkage generally separate the observations into their correct groups.

Single linkage singletons However, single linkage identifies one point as belonging to its own cluster. A more sensible answer is obtained when four clusters are selected, although there are still two singletons.

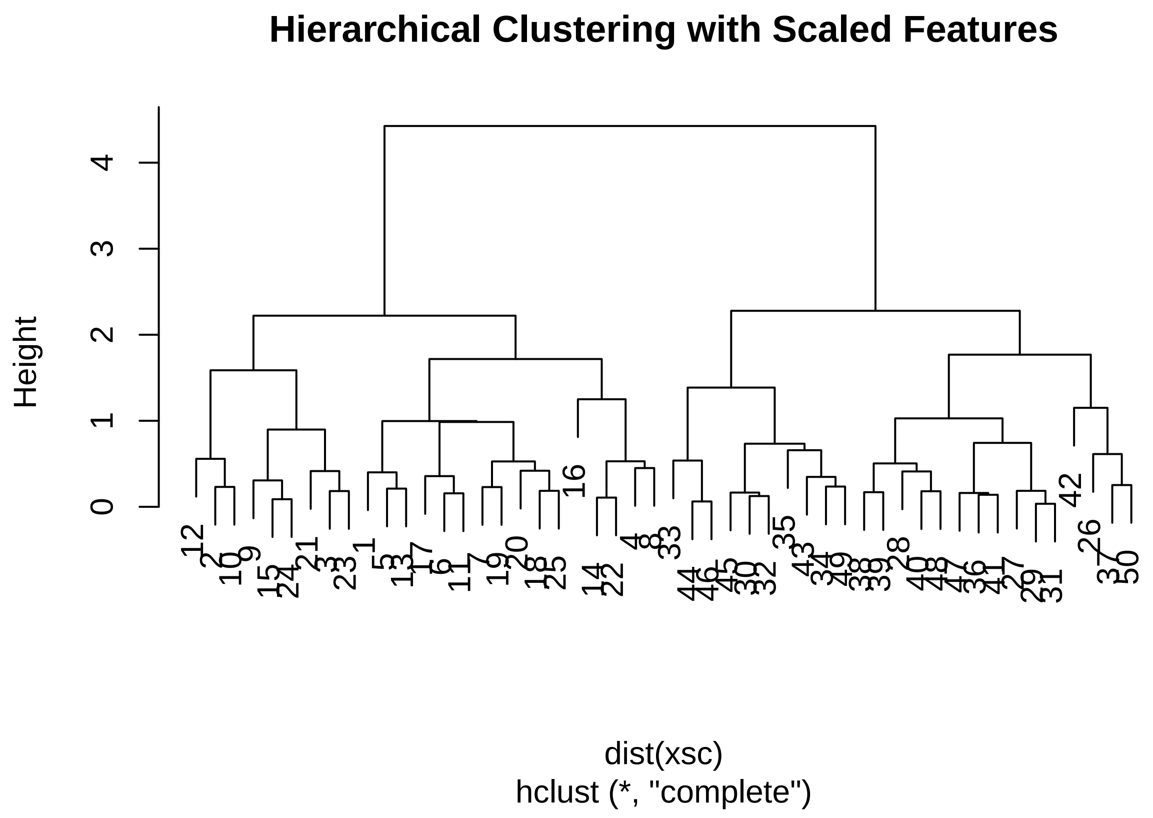
```r
cutree(hc.single, 4)
```

```
##  [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
## [36] 3 3 3 3 3 3 4 3 3 3 3 3 3 3 3
```

Scaling the variables

To scale the variables before performing hierarchical clustering of the observations, we use the `scale()` function:

```r
xsc=scale(x)
plot(hclust(dist(xsc), method="complete"), main="Hierarchical Clustering with Scaled Features")
```



Correlation-based distance

Correlation-based distance can be computed using the `as.dist()` function, which converts an arbitrary square symmetric matrix into a form that the `hclust()` function recognizes as a distance matrix.

However, this only makes sense for data with at least three features since the absolute correlation between any two observations with measurements on two features is always 1.

```r
# Code below is to check the statement above.

x=matrix(rnorm(5*2), ncol=2)
dd=as.dist(1-cor(t(x)))
dd
```

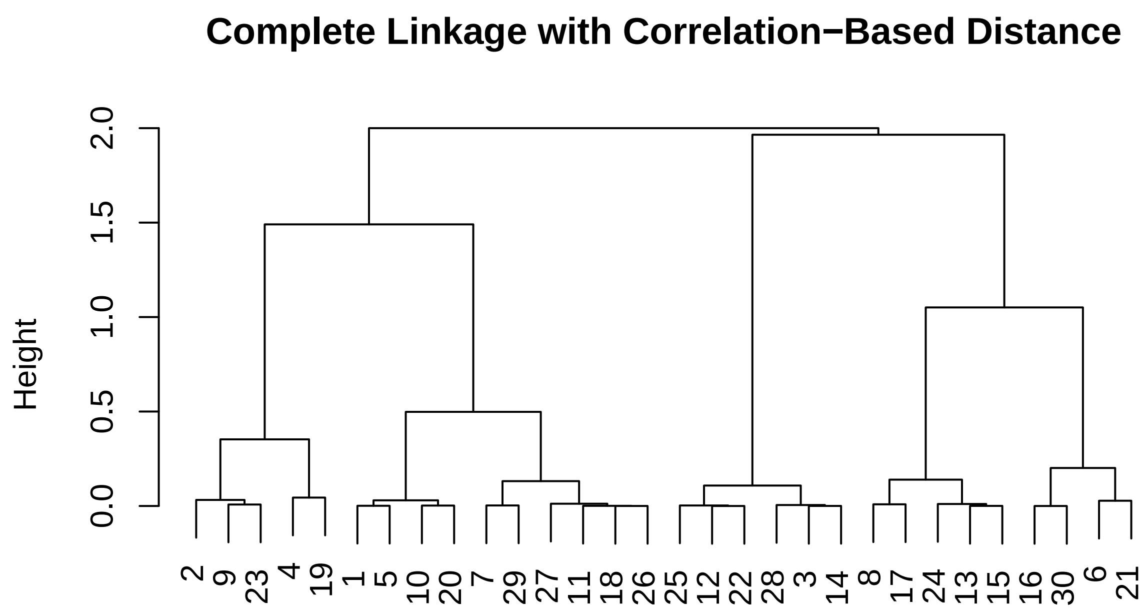
```
##   1 2 3 4
## 2 2
## 3 2 0
## 4 2 0 0
## 5 0 2 2 2
```

```
x=matrix(rnorm(5*3), ncol=3)
dd=as.dist(1-cor(t(x)))
dd
```

```
##              1            2            3            4
## 2 1.765076e+00
## 3 1.638313e+00 1.595186e-02
## 4 1.630165e+00 1.788151e-02 5.554177e-05
## 5 1.954130e+00 7.722732e-02 1.605010e-01 1.662744e-01
```

Hence, we will cluster a three-dimensional data set.

```
# Simulate data
x=matrix(rnorm(30*3), ncol=3)
# Compute correlation-distance matrix
dd=as.dist(1-cor(t(x)))
# Plot dendogram
plot(hclust(dd, method="complete"),
     main="Complete Linkage with Correlation-Based Distance",
     xlab="", sub="")
```


Complete Linkage with Correlation−Based Distance
Height