Chapter 8

Cluster Analysis

8.1 Introduction

Cluster Analysis is the most well-known example of *unsupervised learning.* Unsupervised learning means that we do not have a training set of examples where the classification is known; we learn the number of the classes and assign objects to classes purely using the data. It is a tool for arranging large quantities of multivariate data into natural groups.

Consider a sample of n independent observations on p variables, $(x_{j1}, \dots, x_{jp})_{j=1}^{n}$. The data may come from various groups, but the number or size of each group may not be known in advance. The idea of cluster analysis is to derive, from the data, the characteristics of the various groups from which the outcomes in the sample came.

Absolutely nothing is assumed about the data, except that the n observations come from K groupings (where K is not known in advance), where for group j, $\mathbb{E}[\underline{X}] = \underline{\mu}_j$. The number K and the vectors $\underline{\mu}_j$, $j = 1, \dots, K$ have to be estimated from the data through cluster analysis. In other words, cluster analysis only attempts to locate the collection of different average values and assigns the observations to the groups determined by these averages.

Cluster analysis techniques are purely numerical, measuring distances and assigning observations to groups based on the distance measures; the techniques do not involve statistics. On the one hand, the techniques do not seem to be very powerful; in a few cases, where the data seems to have some very pronounced centres, cluster analysis techniques can locate them. Having said that, they can work well for large data sets.The important advantage of clustering is that very few modelling assumptions are required.

There are numerous ways of clustering a data set of n independent observations on p correlated variables. We consider three:

1. **Clustering Observations** This is the usual use of the term 'clustering'; we divide the observations into K groups.

2. **Clustering Variables** In situations where there are large numbers of variables, many of which

are highly correlated with each other (for example gene expression levels), We may wish to partition the p *variables* into K distinct groups. For each cluster of variables, a *representative* variable is taken and used.

3. **Two-way clustering** In some situations, both variables and observations may be clustered *together*. For example, we may want to cluster both genes and tissue samples at the same time to see which subset of genes is most closely related to which subset of tissue samples.

In this lecture, we concentrate on the first of these tasks, clustering *observations*.

Warning In many situations, cluster analysis is a total waste of time; the algorithms will only detect the simplest patterns. Consider, for example, a bivariate data set, coming from two random variables; the first uniformly distributed in the circle of radius 1 and the second uniformly distributed in the annulus $\{(x, y)|1.5 \leq x^2 + y^2 \leq 2.5\}$. A cluster analysis is unlikely to detect that there are two different groups.

8.2 Distance and Dissimilarity Measures

Let x_{ik} denote observation i on variable k, where there are n observations on p variables. Usually, *quantitative* variables are *standardised* before the analysis;

$$s_k = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(x_{jk} - \bar{x}_{.k})^2}$$

and

$$y_{ik} = \frac{x_{ik} - \bar{x}_{.k}}{s_k}.$$

For quantitative variables, a common distance measure in cluster analysis is the *Euclidean distance*:

$$d_{ij} = \sqrt{\sum_{k_1}^{p}(y_{ik} - y_{jk})^2}.$$

The statistical distance between two p-variate observations $\underline{x}$ and $\underline{y}$ is usually of the form

$$d(\underline{x}, \underline{y}) = \sqrt{(\underline{x} - \underline{y})^t S^{-1}(\underline{x} - \underline{y})}$$

where S is the sample covariance matrix. This is the simply the *Mahalanobis* distance. The Penrose distance may be used instead. There are many choices of distance.

In situations where the data is categorical, such a distance measure is not meaningful. It is then useful to try and compare data by the presence and absence of certain characteristics in the p variables. The distance is then the number of differences. For example, suppose $p = 5$ and the five variables for items i and k are coded as

	1	2	3	4	5
i	1	0	0	1	1
k	1	1	0	1	0

Then the distance $d(i, k) = 2$.

In some cases, a $1-1$ match is stronger evidence of similarity than a $0-0$ match. For example, if two people can understand ancient Greek, this is stronger evidence of a similarity than if both of them do not read ancient Greek. Then the following table is useful:

$i\backslash k$	1	0	
1	a	b	$a+b$
0	c	d	$c+d$
	$a+c$	$b+d$	$p=a+b+c+d$

This leads to the definition of *similarity*, which is a number that lies between 0 and 1, equal to 1 if the two objects are identical and equal to 0 if the two objects have nothing in common. The following are common similarity measures that are used depending on the situation.

1. $s_{ik} = \frac{a+d}{p}$ if equal weight is given to $1-1$ and $0-0$ matches.
2. $s_{ik} = \frac{2(a+d)}{2(a+d)+b+d}$ if double weight is given to $1-1$ matches than $0-0$ matches
3. $s_{ik} = \frac{a}{p}$ if the $0-0$ matches do not contribute to similarity.
4. $s_{ik} = \frac{a}{a+b+c}$ if the $0-0$ matches are considered irrelevant.
5. $s_{ik} = \frac{2a}{2a+b+c}$ Double weight for $1-1$ matches, $0-0$ matches irrelevant
6. $s_{ik} = \frac{a}{a+2(b+c)}$ $0-0$ matches irrelvant, double weight for unmatched pairs.
7. $s_{ik} = \frac{a}{a+b}$ Ratio of matches to mis-matches, the $0-0$ matches excluded.

The *dissimilarity* is defined as $d_{ik} = 1 - s_{ik}$.

8.3 Clustering Techniques

We first consider *hierarchical* metohds for clustering. These methods start by computing the distances, or dissimilarities (depending on the type of data) from each object to every other object in the sample and storing this as a dissimilarity matrix. Groups are then formed by a process of *agglomeration* or *division*. Agglomeration techniques start with all individuals in groups of size 1. Close groups are merged until the process is complete. With division, all objects start in a single group. This group is split into two and then the subgroups are split into pairs until all the elements in each group are sufficiently similar and the groups as a whole are sufficiently far apart from each other.

There is also a *partitioning* approach to cluster analysis. For algorithms that fall into this class, objects are permitted to move in and out of groups at various stages in the analysis. First, some more or less arbitrary group centres are chosen. Objects are allocated to the nearest group centre. New centres are then calculated, representing the averages of the objects in the groups. An object is then moved to a new group if it is closer to that group's centre than the centre of the present group. Groups that are close together are merged; groups that are spread out are split, following some defined rules. The process continues iteratively until stability is achieved with a predetermined number of groups. For each run, the number of groups is fixed in advance. This is repeated over a range of values for the group number and the output is the most successful one, where some criteria for goodness of fit are given.

8.4 Hierarchic Methods

First, a matrix of distances between each object is calculated. For example, if there are 5 objects, the matrix might be

	object				
object	1	2	3	4	5
1	–				
2	2	–			
3	6	5	–		
4	10	9	4	–	
5	9	8	5	3	–

Firstly, each object is considered to be in a group on its own. Then run through all possible distances, from lowest to highest. As the distance is increased, a criterion for deciding whether groups should merge is decided upon.

Single Linkage With *single linkage*, also known as *nearest neighbour linkage*, the groups are merged as the distance increases if one of the objects in a group has distance less than or equal to that distance from an object in another group. In the example above, no merging takes place for a distance less than 2. At 2, (1) and (2) merge to form a new group, $(1, 2)$. The groups, formed according to *nearest neighbour linkage*, are given below, as the distance increases.

distance	groups
$0, 1$	$(1), (2), (3), (4), (5)$
2	$(1, 2), (3), (4), (5)$
3	$(1, 2), (3), (4, 5)$
4	$(1, 2), (3, 4, 5)$
5	$(1, 2, 3, 4, 5)$

This may be expressed as a *dendrogram*.

Complete Linkage With *complete linkage*, also known as *furthest neighbour linkage*, two groups merge only if the most distant members of the two groups are close enough. Starting with each member in a group on its own, the groups formed under furtherst neighbour linkage, as the distance is increased and using the groups already formed, are given below.

distance	groups
$0, 1$	$(1), (2), (3), (4), (5)$
2	$(1, 2), (3), (4), (5)$
3	$(1, 2), (3), (4, 5)$
5	$(1, 2), (3, 4, 5)$
10	$(1, 2, 3, 4, 5)$

As with all hierarchical techniques, the dendrogram is a natural way to visualise it graphically. With *group average linkage*, two groups merge if the *average* distance between them is small enough. Merging groups based on *average* distance, letting the distance range through the continuum, yields

distance	groups
0	$(1), (2), (3), (4), (5)$
2	$(1, 2), (3), (4), (5)$
3	$(1, 2), (3), (4, 5)$
4.5	$(1, 2), (3, 4, 5)$
7.8	$(1, 2, 3, 4, 5)$

Average Linkage Again, the input into the algorithm may be distances or dissimilarities. The algorithm begins by searching for the closest objects. Suppose these are U and V. These are then merged to form a cluster (UV).

Now, suppose that (UV) and W are clusters. The distance between the two clusters is defined as

$$d_{(UV)W} = \frac{1}{N_{(UV)}N_W} \sum_{ik} d_{ik}.$$

The Ward Clustering Algorithm

The Ward Clustering Algorithm does not measure the distance between groups in terms of nearest neighbours, furthest neighbours or group centres. Rather, it tries to join groups that do not increase a given measure of heterogeneity too much. That is, the distance is related to the *variance* within the group; two groups are merged if the within-group variance of the merged group is lower than a specified level.

The Ward Clustering algoritm uses the following distance function: let the distance between groups containing *single* multivariate observations $\underline{x}$ and $\underline{y}$ be

$$d(\underline{x}, \underline{y}) = \sqrt{\sum_{j=1}^{p}(x_j - y_j)^2}.$$

Then, for larger groups, the distance between them is computed inductively. If two *distinct* groups P and Q are merged to form a group $P \cup Q$, the distance between $P \cup Q$ and another group R, in terms of the previously calculated $d(P, R)$ and $d(Q, R)$ is given by

$$d(P \cup Q, R) = \frac{n_R + n_P}{n_R + n_P + n_Q} d(P, R) + \frac{n_R + n_Q}{n_R + n_P + n_Q} d(Q, R) - \frac{n_R}{n_R + n_P + n_Q} d(P, Q),$$

where n_R, n_P, n_Q are the numbers in R, P, Q respectively Two groups are merged when the Ward distance between them reaches the specified level.

Motivation The *inertia* of a group R is defined as

$$I_R = \frac{1}{n_R} \sum_{j=1}^{n_R} d^2(\underline{x}^{(j)}, \bar{\underline{x}}_R);$$

in other words, the inertia is the average squared distance from each group member to the group centre. When two distinct groups P and Q are merged, the new group $P \cup Q$ has a larger inertia than the sum of inertias of P and Q. That is, if P and Q are disjoint, then a straightforward computation yields that:

$$I_{P \cup Q} \geq I_P + I_Q.$$

More precisely, if d is the Ward measure, then a straightforward computation gives the following expression for $I_{P \cup Q} - I_P - I_Q$:

$$\Delta(P, Q) := I_{P \cup Q} - (I_P + I_Q) = \frac{n_Q n_P}{n_Q + n_P} d^2(P, Q).$$

With the Ward algorithm, therefore, the next pair of groups to be joined as the distance parameter increases is the pair of groups that currently gives the smallest value for $\Delta(P, Q)$.

8.4.1 Agglomerative Clustering in R

We'll see whether or not a cluster analysis on the European Employment Data can recover the four categories.

- step 1: standardise the nine variables.
- step 2: compute the Euclidean distances between *all* pairs of countries, using the standardised variables.
- step 3: construct a dendrogram using, for example, the agglomerative, nearest neigbour, hierarchic process.

```
> www =
"https://www.mimuw.edu.pl/~noble/courses/MultivariateStatistics/data/
employment.csv"
> employment <- read.csv(www,header=T)
> y<-scale(employment[,3:11])
> emp<-employment
> emp[,3:11]<-y
```

The 'Ward' method may be implemented as follows:

```
> d <- dist(y, method = "euclidean")
> fit <- hclust(d, method="ward.D2")
> plot(fit,labels=employment$group)
> rect.hclust(fit, k=4, border="red")
```

The plot is shown in figure **??**.
At a distance of about 1, there are four clusters. The performance of the clustering algorithm, using the Ward distance is clear from the plot. Other distance measures give less satisfactory results.

8.5 Divisive Analysis (`diana`)

Divisive hierarchic methods are less common. Objects start in a single group. Then, the object furthest from the mean is split off. Then objects from the main group are moved to the new group if they are closer to the new group as it stands, than from the main group. The pair of groups from the original group has been established when all objects are closest to the centre of the group they are in.The most-used divisive hierarchical clustering procedure was proposed by MacNaughton-Smith, Williams, Dale, and Mockett (1964).

Suppose the algorithm has reached a stage where there are several clusters. Let the current collection of clusters be $\mathcal{C}$. The cluster $C \in \mathcal{C}$ that has the largest average dissimilarity between an item and the remaining items in the cluster is chosen.

A *splinter* group (say cluster A) is computed, where $A \subset C$. The splinter group is initiated by extracting the item that has the largest average dissimilarity from all the other items in A.

Let $B = C \backslash A$. For each of the items in B, we compute (1) the average dissimilarity between that item and all the other items in B and (2) the average dissimilarity between that item and all the items in A. We compute the difference (1)-(2) for each item in B. If all the differences are negative, we stop. If any of the differences are positive, we move the item with the largest (1)-(2) into A and then repeat until we have a division of C into two clusters, A and B. This is continued recursively; after each division, the largest average distance between each item and the other items in the cluster is recorded.

8.6 Non-hierarchical Clustering Methods

Non-hierarchical methods start from either (1) an initial partition of items into groups or (2) an initial set of seed points, which will form the nuclei of clusters. One way to start is to randomly select seed points from among the items or to randomly partition them into groups. This eliminates bias.

8.6.1 K-means method

This algorithm proceeds as follows:

1. Partition the items into K initial clusters.
2. Proceed through the list of items assigning an item to the cluster whose mean is nearest. Distance is usually computed using Euclidean measure with standardised variables. Recalculate the mean for the cluster receiving the new item and for the cluster losing the item.
3. Repeat step 2 until no more reassignments take place.

Example 8.1.

Consider 4 measurements, A, B, C, D of the two variables (X_1, X_2). The data is

	x_1	x_2
A	5	3
B	-1	1
C	1	-2
D	-3	-2

Suppose they are to be divided into two clusters. To start with, *randomly* assign them to two clusters. For example, suppose this were AB and CD. Then the co-ordinates for the cluster centres (means) are

$$(AB) : (\bar{x}_1, \bar{x}_2) = (\frac{5 + (-1)}{2}, \frac{3 + 1}{2}) = (2, 2)$$

$$(CD) : (\bar{x}_1, \bar{x}_2) = (\frac{1 + (-3)}{2}, \frac{-2 + (-2)}{2}) = (-1, -2).$$

Using Euclidean distance, the distance from each item to each group centre is computed and then the group centre updated to:

$$\bar{x}_{i,new} = \begin{cases} \frac{n\bar{x}_i + x_{ji}}{n+1} & \text{item j added} \\ \frac{n\bar{x}_i - x_{ji}}{n-1} & \text{item j removed} \\ \bar{x}_i & \text{no change} \end{cases}$$

In the first round, if A were re-assigned to group (ACD), the group centres would then be

$$\underline{\bar{x}}_{ACD} = (1, -\frac{1}{3}), \quad \underline{\bar{x}}_B = (-1, 1).$$

Since A is closer (using Euclidean distance) to the centre of AB than to ACD, it is not moved. By proceeding, the final partition is (A), (BCD).

8.6.2 K-medoids

A *medoid* is simply a representative object. The *K-medoids algorithm* searches for K 'representative objects' rather than the centroids. A dissimilarity-based distance is used instead of squared-Euclidean distance. Because it minimises a sum of dissimilarities instead of a sum of squared Euclidean distances, it is more robust to data anomalies such as outliers and missing values.

K-medoids starts with the proximity matrix D and an initial configuration into K clusters. The *representative object* is the object that minimises the total dissimilarity to all other items in the cluster. From this point onwards, the role of centroids in K-means is replaced by represenative objects in K-medoids and the sum of squares of Euclidean distances replaced by the sum of squares of dissimilarities. With these changes, the algorithm proceeds in the same way.

8.6.3 Partitioning Around Medoids (pam)

This clustering method is a modification of the *K-medoids algorithm.*The `pam` algorithm modifies K-medoids by introducing a swapping strategy by which the medoid of a cluster is replaced by the item in the cluster that minimises the value of an objective function that may be different from the sum of squares of the dissimilarities.

K-medoids and `pam` run well on small data sets, but are not efficient enough to use for clustering large data sets.

8.6.4 Silhouette Plot

With `kmeans` and `pam`, the result of the partition can be represented graphically in a so-called *silhouette plot.* Suppose the data is split into K clusters. Let $c(i)$ denote the cluster of item i. Let c be some cluster different from $c(i)$ and $d(i, c)$ the average dissimilarity between i and items of c. We compute $d(i, c)$ for all clusters other than $c(i)$. let

$$b_i = \min_{c \neq c(i)} d(i, c)$$

and a_i the average dissimilarity between i and all items of the same cluster. The ith silhouette value is

$$s_{iK} = \frac{b_i - a_i}{\max a_i, b_i}.$$

Large positive values indicate that item i is well clustered, large negative that it is badly clustered. A *silhouette plot* is a plot of these after they have been ranked in decreasing order for each cluster, where the length of the bar is s_{iK}.

The average silhouette width may also be used to find the best value of K, by maximising $\overline{s}_K$. The *silhouette coefficient* is $s_C = \max_K \overline{s}_K$.

8.7 Self Organising Maps (SOM)

The *self-organising map* algorithm is due to Kohonen (1982). The basic idea is *data reduction.* A set of n observations in (say) $\mathbb{R}^p$ is reduced to a number $K = K_1 \times K_2$ nodes, where $K << n$. These nodes are represented on a 2-d grid. To each node m is assigned a *representative value* $x_m \in \mathbb{R}^p$, or whatever the state space for the observations is. Observations are assigned to a node m if they lie within the neighbourhood of the representative value x_m. These nodes may be considered as nodes of a neural network and there is a link between two nodes a and b if their representative values lie within each others neighbourhood.

Two versions of the algorithm are available; the *on-line* version and the *batch* version. The end product of SOM, after a large number of iterative steps, is a graphical image known as an SOM plot. This is displayed in output space and consists of a grid (or network) of a large number of inter-connected *nodes* or artificial neurons. The SOM algorithm has much in common with K-means.

8.7.1 On-Line Version

For a 2-d SOM, firstly set up the map size $K_1 \times K_2$ (select K_1 and K_2 usually larger than intended for the final output). Let $\mathcal{K}$ denote the space of labels; $k = (i, j) \in \mathcal{K}$ for $i \in \{1, \ldots, K_1\}$ and $j \in \{1, \ldots, K_2\}$. For each $k \in \mathcal{K}$, choose at random a representative value $m_k \in \mathbb{R}^p$ (or whatever the input space is) for the standardised data matrix.

Firstly, the data set is standardised so that the columns have mean 0 and variance 1. At each stage, an input vector X is randomly selected and assigned to k^* where

$$k^* = \text{argmin}_k \|X - m_k\|.$$

k^* is declared to be the winning node. We then look at nodes that are 'neighbours' of the winning node. Two nodes k and k' are neighbours if the Euclidean distance between m_k and $m_{k'}$ is less than a given threshold c. We update the value of m_k for the winning node and for all its neighbours using:

$$m_k \leftarrow m_k + \alpha(X - m_k) \qquad k \in N(k^*).$$

Here $\alpha \in (0,1)$ is the *learning factor.*

A useful rule of thumb is to run the algorithm for $500 \times K_1 \times K_2$ steps.

A *distance weighted* strategy is more popular:

$$m_k \leftarrow m_k + \alpha h_k (X - m_k)$$

where

$$h_k = \exp\left\{ \frac{-\|m_k - m_{k^*}\|^2}{2\sigma^2} \right\} \mathbf{1}_{\{k \in N_c(k^*)\}}$$

where $N_c(k)$ denotes the neighbourhood of k with parameter c. Values of c, α and σ are provided by the user, but may change during the process.

Batch Version m_k is updated by listing all items X_i already examined whose $m_{k^*} \in N_c(k)$. These are averaged over and the average is used for the update. This turns out to be much faster.

8.8 Clustering Based on Mixture Models

Suppose that there are K population types and that an observation from population j is distributed according to density function f_j; there is a proportion p_j of population j in the total population. Then an observation taken at random is from the *mixing distribution*

$$f(\underline{x}) = \sum_j p_j f_j(\underline{x}).$$

For example, an individual from population j is taken from a $N(\underline{\mu}_j, C_j)$ population.

Inferences are then based on the likelihood, which for N objects and a fixed number K of clusters is

$$L(p_1, \ldots, p_K, \underline{\mu}_1, \ldots, \underline{\mu}_K, C_1, \ldots, C_K) = \prod_{i=1}^{N} f(\underline{x}_i | \underline{\theta})$$

where $\underline{\theta}$ denotes the entire parameter collection, including the class label.

When the number K is fixed, the approach is to choose the parameters, which for a mixture of normals are $\widehat{p}_j$, $\widehat{\underline{\mu}}_j$, $\widehat{C}_j$ for $j = 1, \ldots, K$.

Clearly, the more parameters, the better the apparent fit; if the number of parameters and number of data points are equal, the fit will be perfect, but this will give us no information.

When the number of clusters is not set in advance, the aim is to compare values of

$$\log L_{\max}(k \text{ clusters}) - \text{penalty}(k)$$

where the penalty increases with the number of parameters introduced into the model. There are several common approaches. One is the Akaike Information Criterion (AIC), where the penalty is $2N \times$ (number of parameters) so that

$$AIC = 2\log L_{\text{max}}(k \text{ clusters}) - k2N(\frac{1}{2}(p+1)(p+2) - 1)$$

There are $k \times p$ parameters for mean values, $k \times \frac{p(p+1)}{2}$ covariance parameters (k clusters, $\frac{p(p+1)}{2}$ parameters for each covariance matrix, $k-1$ parameters for the proportion of population in each cluster, because $\sum_{j=1}^{k} p_j = 1$. The Bayesian Information Criterion is similar, but uses $\log N$;

$$BIC = 2\log L_{\text{max}}(k \text{ clusters}) - 2(\log N)(K\frac{1}{2}(p+1)(p+2) - 1)$$

Warning: the maximum likelihood technique in this context is unsound Much of the 'general theory' surrounding Maximum Likelihood techniques is connected with exponential families. As soon as one leaves the framework of exponential families, properties such as consistency have to be proved on a case by case basis and often they fail. An example of relevance here: let $x_1, \ldots, x_n$ be a random sample from a distribution with density

$$g(x) = \frac{1}{10}\phi(x-\mu) + \frac{9}{10}\frac{1}{\sigma}\phi\left(\frac{x-\mu}{\sigma}\right).$$

Suppose that $\sigma > 0$. The likelihood function for (μ, σ) is:

$$L(\mu, \sigma; x_1, \ldots, x_n) = \prod_{j=1}^{n} g(x_j|\mu, \sigma)$$

and this is maximised for $(\widehat{\mu}, \widehat{\sigma}) = (x_j, 0)$ for $j \in \{1, \ldots, n\}$, where the value of the likelihood is $+\infty$.

Therefore, in this case, an unconstrained ML approach is guaranteed to produce garbage; a $+\infty$ can be obtained for the likelihood for a ridiculous model (the model where the estimated standard deviation parameter for one of the groups $\widehat{\sigma}$ is 0, no matter what the true value of σ is).

Estimating parameters therefore proceeds by applying suitable constraints.

8.8.1 The E-M Algorithm

A popular algorithm for finding maximum likelihood estimates when there is missing data is the EM (expectation maximisation) algorithm. It proceeds as follows: Let $X = (X_o, X_m)$ where X_o denotes observed data and X_m denotes missing data.

Here, for each observation, the *missing datum* is the class label. For K multivariate normal clusters, the observed data is an observation from the distribution: $X_o|\{X_m = i\} \sim N(\mu_i, C_i)$; we have $\mathbb{P}(X_m = i) = p_i$.

The E-M algorithm proceeds as follows:

1. Inpute $\widehat{\theta}_0$, an initial guess for the parameters θ.
2. For $m = 0, 1, 2, \dots$ iterate between the following two steps:
    - E-step Compute
    $$Q(\theta|\widehat{\theta}_m) = \mathbb{E}_{\widehat{\theta}_m}\left[\log L(\theta; X))|X_o\right]$$
    as a function of θ (or, at least, compute it or estimate it for several candidate values of θ)
    - M-step Find
    $$\widehat{\theta}_{m+1} = \text{argmax}_\theta Q(\theta|\widehat{\theta}_m).$$
    - Stop when convergence of the likelihood function is attained.

Note that the expectation is over the *missing* variables in the observations.

Using $x \in \mathbb{R}^p$ to denote a value in the observation space of the observed variables and f to denote the density function,
For Gaussian clustering, where X_m denotes the class variable and $p_i = \mathbb{P}(X_m = i)$ and there are n observations,

$$Q(\theta|\widehat{\theta}_m) = \sum_{j=1}^{n}\sum_{i=1}^{K}\widehat{p}_i^{(m)} \log \sum_{i=1}^{K} p_i f(x_{jo}|\mu_i, C_i).$$

The parameter vector is $\theta = (\mu_1, \dots, \mu_K, C_1, \dots, C_K, p_1, \dots, p_K)$ with the constraint that $p_i \geq 0$, $\sum_i p_i = 1$, C_j's are positive definite.

While the maximiser is difficult to find directly, it can be carried out in stages. For example, it is a straightforward computation that the values for $p_1, \dots, p_K$ which maximise the above functional are:

$$\widetilde{p}_i = \frac{1}{n}\sum_{j=1}^{n} \frac{\widehat{p}_i^{(m)} f(x_j|\mu_i, C_i)}{\sum_{k=1}^{K} \widehat{p}_k^{(m)} f(x_j|\mu_k, C_k)}.$$

First the parameters $p_1, \dots, p_K$ are updated, then a gradient ascent is used to find updates for $\mu_1, \dots, \mu_K$ and finally the $C_1, \dots, C_K$ are updated.

For EM clustering, once the number of classes, the parameters π_i (probabilities for each class), μ_i (means for each class) and C_i (covariance matrices for each class) have been established, classification then proceeds by, for an observation x, finding the class i that maximises the posterior $\pi_i \widehat{L}_i(x)$ where $\widehat{L}_i$ denotes the estimated likelihood function for class i.

In multivariate Gaussian mixture problems, the curse of dimensionality raises its ugly head, since the number of parameters grows rapidly with the dimension/ Although PCA is often used as a first step to reduce the dimensionality, this may not work for Gaussian mixture models, since a class structure may not be preserved by the principal components.

Furthermore, the EM algorithm may break down as in the univariate example with two classes given earlier, when one of the covariance matrices becomes singular or nearly singular.

EM clustering may be carried out in R using the package **EMCluster**.