10. Clustering Analysis

10.1 Introduction

Until now, we have assumed that the training samples used are labelled data. Procedures that use labelled data are said to be supervised. Now we shall explore *unsupervised* learning, which uses unlabelled samples.

Unsupervised learning does not need supervision of labelled training data sets, making it an ideal machine learning technique for discovering patterns, groupings underlying the data. In unsupervised learning, there are two main topics: clustering analysis and association analysis. In this course, we focus on clustering analysis, which has been widely used in market segmentation, search result grouping etc.

10.2 Basics of Clustering Analysis

Roughly speaking, a clustering procedure yield a data description in terms of clustering or grouping of data points that possess strong internal similarities. Formal clustering procedures use a criterion function and seek the grouping that optimises the criterion function.

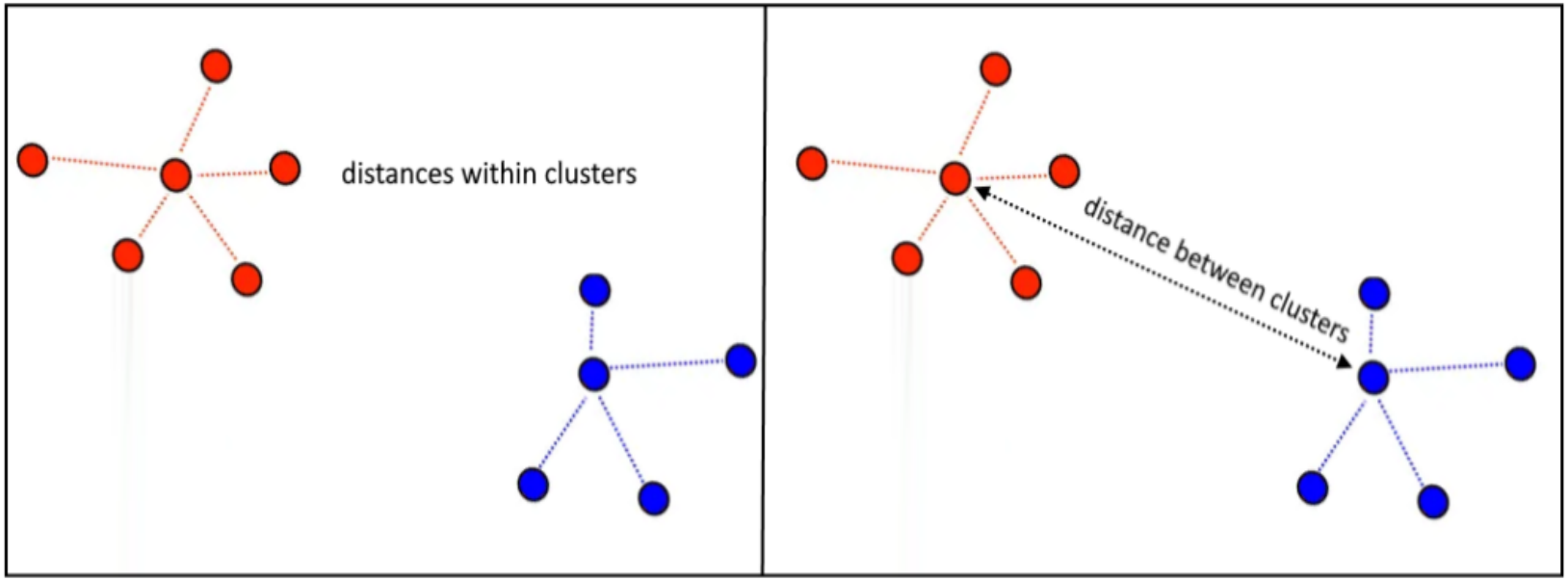
10.2.1 Similarity measure

Once we describe the clustering problem as one of finding natural groupings in a set of data, we need to define:

(1) What we mean by a natural grouping?

(2) In what sense, the samples in one cluster/group are more like one another than like samples in other clusters?

The most obvious measure of the similarity between two samples is the distance between them. It is expected that the distance between samples in the ***same*** cluster to be significantly less than the distance between samples in ***different*** clusters as shown below.



One broad class of distance metrics is of the form:

$$d(\mathbf{x}, \mathbf{x}') = \left(\sum_{i=1}^{d} |x_i - x_i'|^q \right)^{1/q}$$

Where $q \geq 1$ is a selectable parameter. This distance measure is called Minkowski distance.

If $q = 2$, then the distance metric is the commonly used Euclidean distance.

If $q = 1$, then the distance metric is called Manhattan or city block distance.

Some other distance metrics such as Mahalanobis distance and cosine similarity measure can also be used.

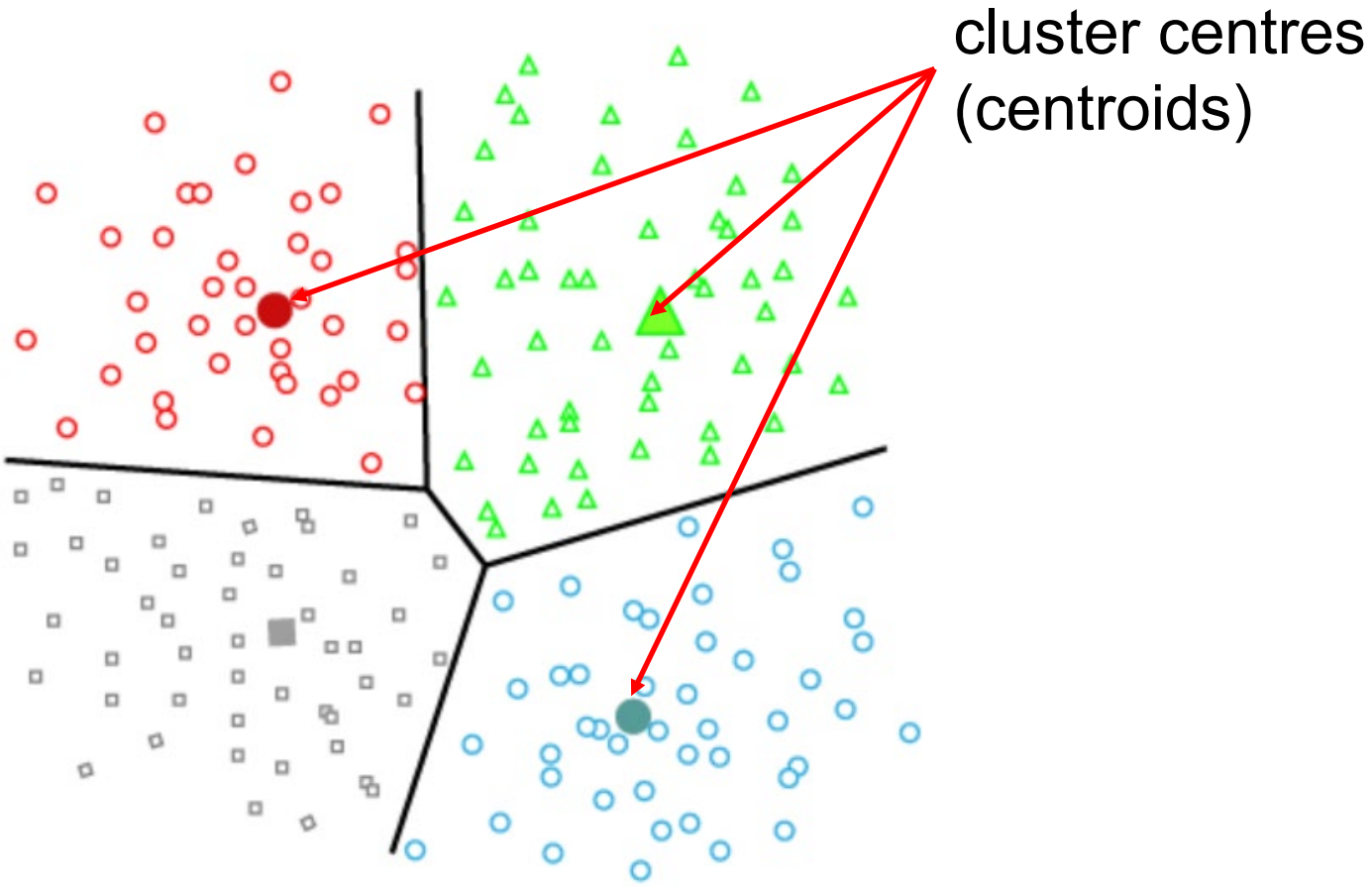
10.2.2 Types of Clustering Algorithms

There are different types of clustering algorithms, including

(1) Centroids-based clustering (partitioning methods)
(2) Hierarchical clustering
(3) Distribution-based clustering
(4) Density-based clustering

Centroid-based method

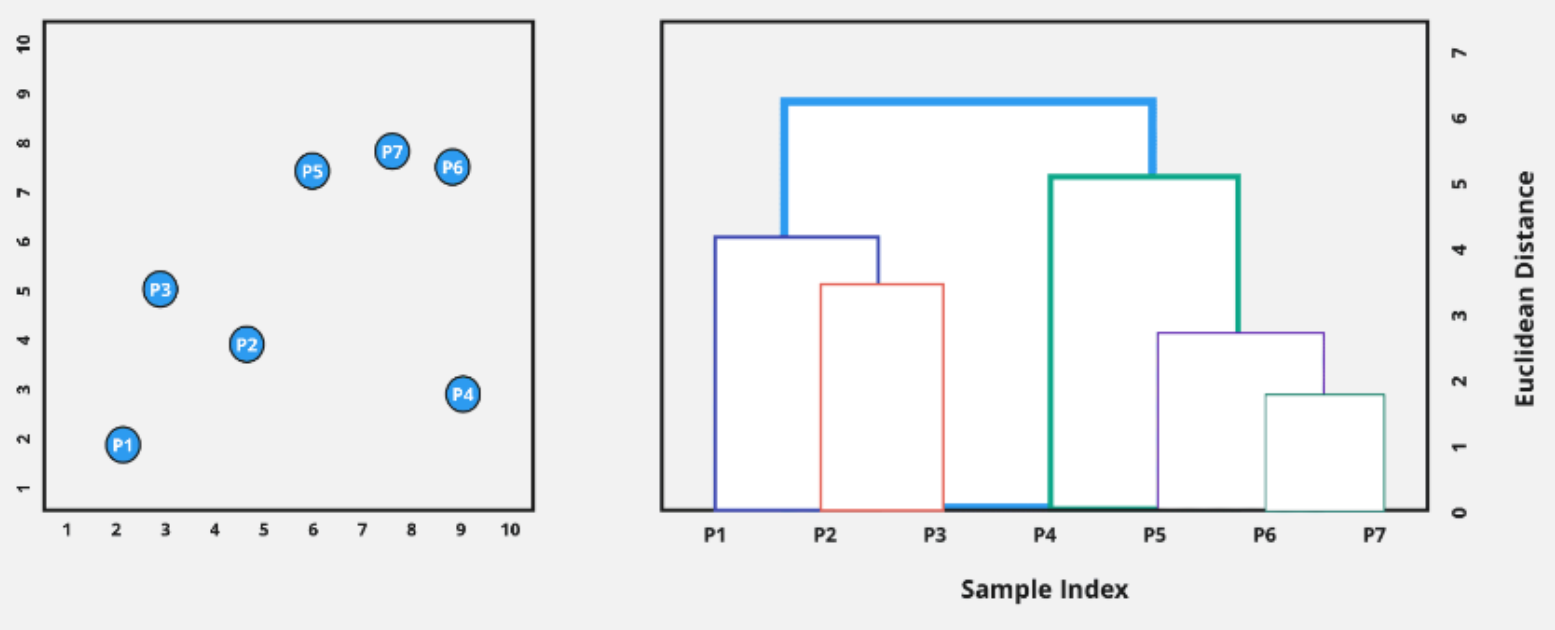
Centroid-based method is also known as partitioning method. It is a type of clustering that divides the data into non-hierarchical groups. The most common example of partitioning clustering is the k-means clustering algorithm.



Partitioning method

Hierarchical Clustering

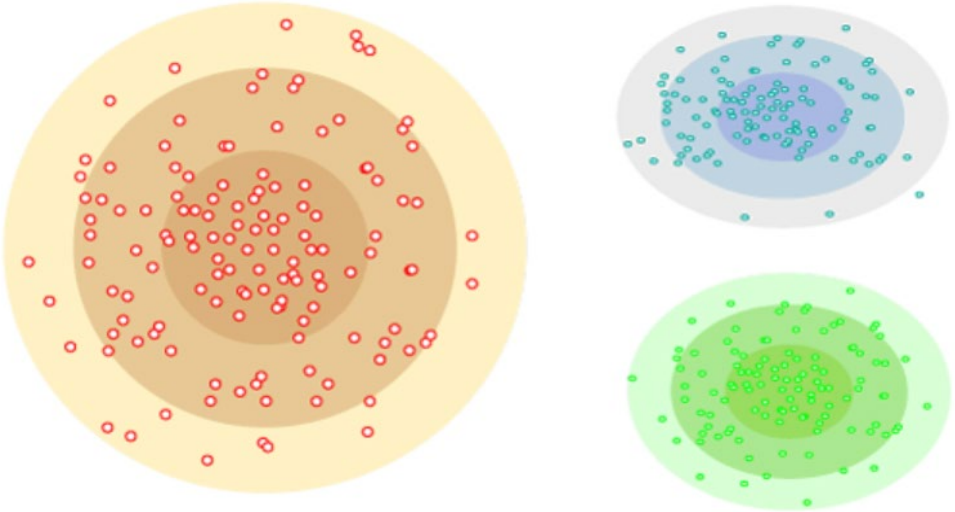
Hierarchical clustering can be used as an alternative for the partitioned clustering as there is no requirement of pre-specifying the number of clusters to be created. In this technique, the dataset is divided into clusters to create a tree-like structure, which is also called a **dendrogram**.



Distribution-based clustering

In the distribution-based clustering method, the data is divided based on the probability of how a dataset belongs to a particular distribution. The grouping is done by assuming some distributions, e.g. Gaussian distribution.

The example of this type is the Gaussian Mixture Models (GMM).

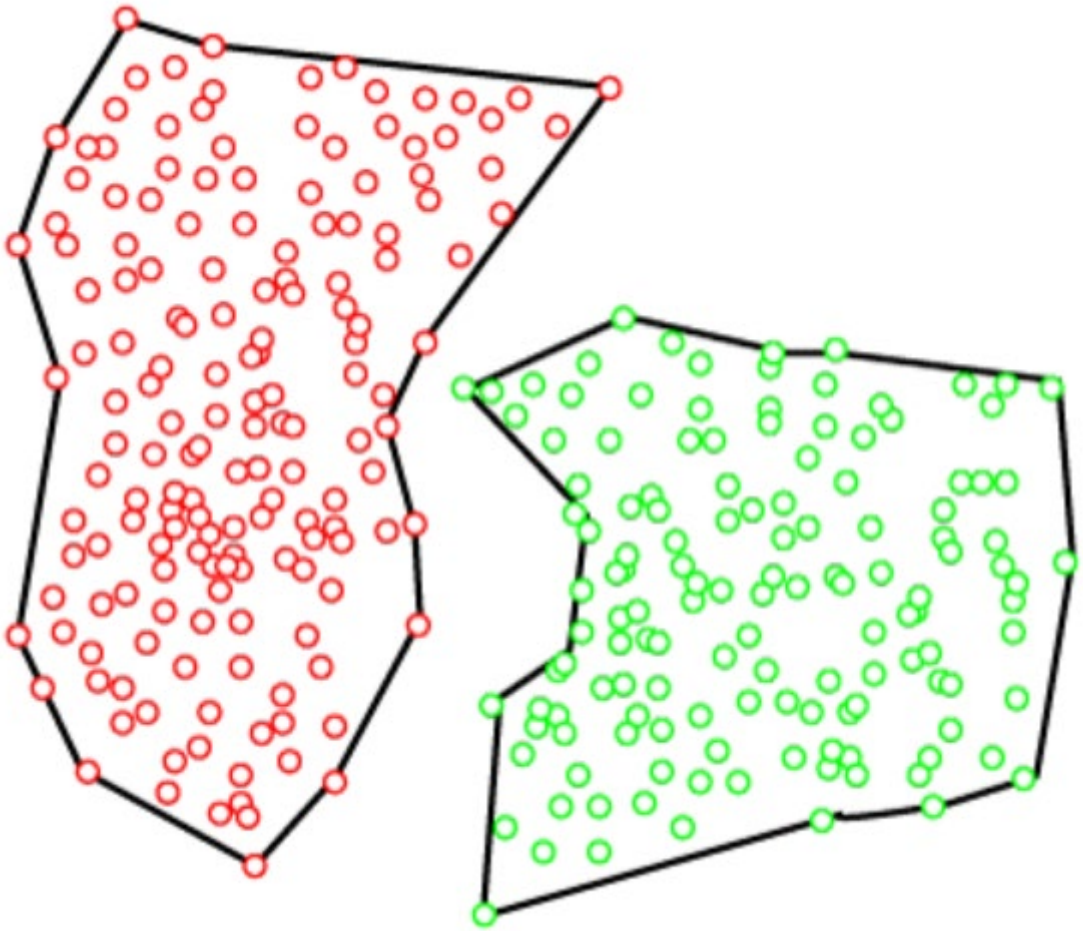


Density-based clustering

The density-based clustering method connects the highly-dense areas into clusters, and the arbitrarily shaped distributions are formed as long as the dense region can be connected.

This algorithm does it by identifying different clusters in the dataset and connects the areas of high densities into clusters. The dense areas in data space are divided from each other by sparser areas.

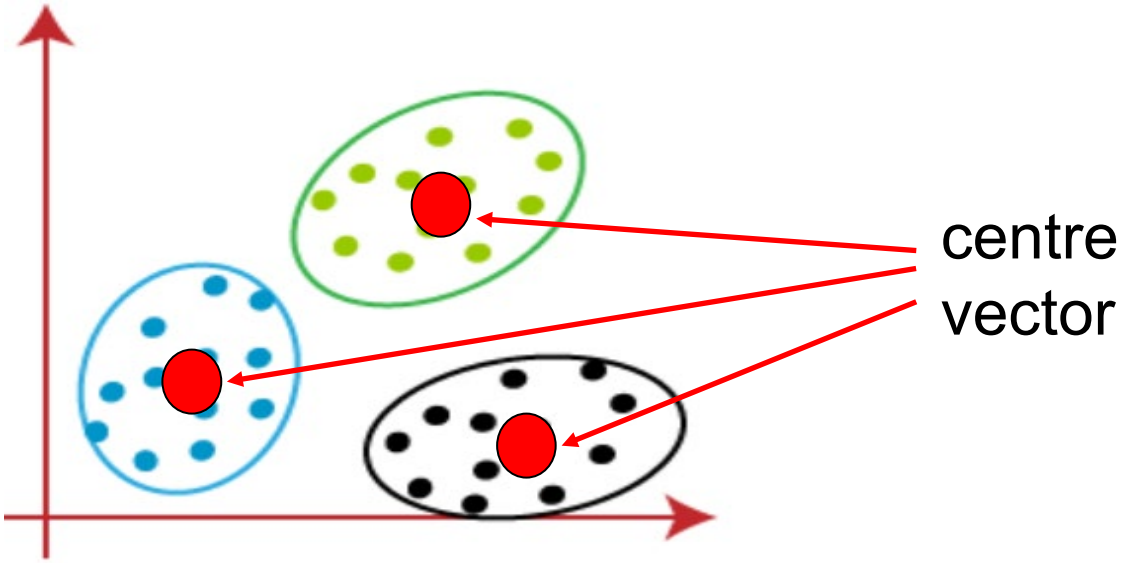
The algorithms can face difficulty in clustering the data points if the dataset has varying densities and high dimensions.



Arbitrarily shaped distributions

10.3 Centroids-based Clustering

Centroid-based clustering is considered as one of the simplest yet effective clustering algorithms. The intuition behind centroid based clustering is that a cluster is characterized and represented by a **centre vector** and data points that are in close proximity to these central vectors are assigned to the respective clusters.



k-means clustering algorithm

K-mean clustering is the most popular centroids-based method.

Suppose that we have a set *D* of *n* samples, $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$, that we want to partition into exactly *k* disjoint subsets D_1, D_2, $\cdots$, D_k. Each subset is to represent a cluster, with samples in the same cluster being somehow more similar than samples in different clusters.

Then the problem is to find the partition that extremizes the criterion function. Next, we examine the characteristics of the criterion function, and later how to find an optimal partition.

Within Cluster Sum of Squares Criterion (WCSS)

K-means clustering adopts the Within Cluster Sum of Squares (WCSS) as the criterion. Let n_i be the number of samples in D_i and let $\mathbf{m}_i$ be the mean of those samples:

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \mathbf{x}$$

Then the Within Cluster Sum (WCSS) of Squares is defined as:

$$WCSS = \sum_{i=1}^{k} \sum_{\mathbf{x} \in D_i} \|\mathbf{x} - \mathbf{m}_i\|^2$$

This criterion function has a simple interpretation: for a given cluster D_i : the mean vector $\mathbf{m}_i$ is the best representative of the samples in D_i in the sense that it minimizes the sum of the squared "error" vectors $\mathbf{x} - \mathbf{m}_i$ in D_i. Thus, $WCSS$ measures the total squared error incurred in representing the *n* samples $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$ by the *k* cluster centres $\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_k$.

The value of $WCSS$ depends on how the samples are grouped into clusters and the number of clusters. The optimal partitioning is defined as one that minimizes $WCSS$.

What kind of clustering problems are well suited to $WCSS$ criterion? Basically, it is an appropriate criterion when the clusters form compact clouds that are rather well separated from one another as shown next.

k-means clustering works

k-means clustering does not work

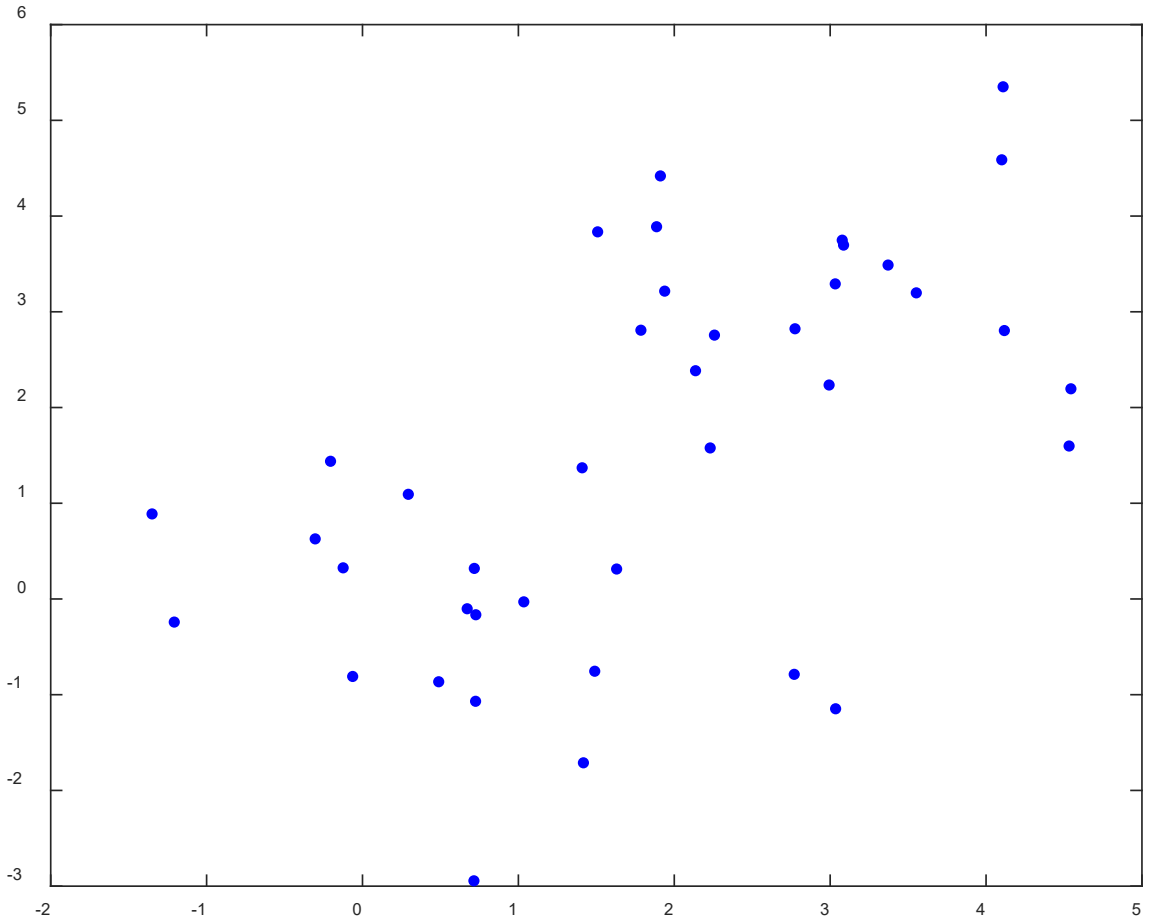
The k –mean clustering algorithm consists of iterations of two main operations:

(1) Assign the samples to the nearest cluster
(2) Compute the new cluster centres (centroids)

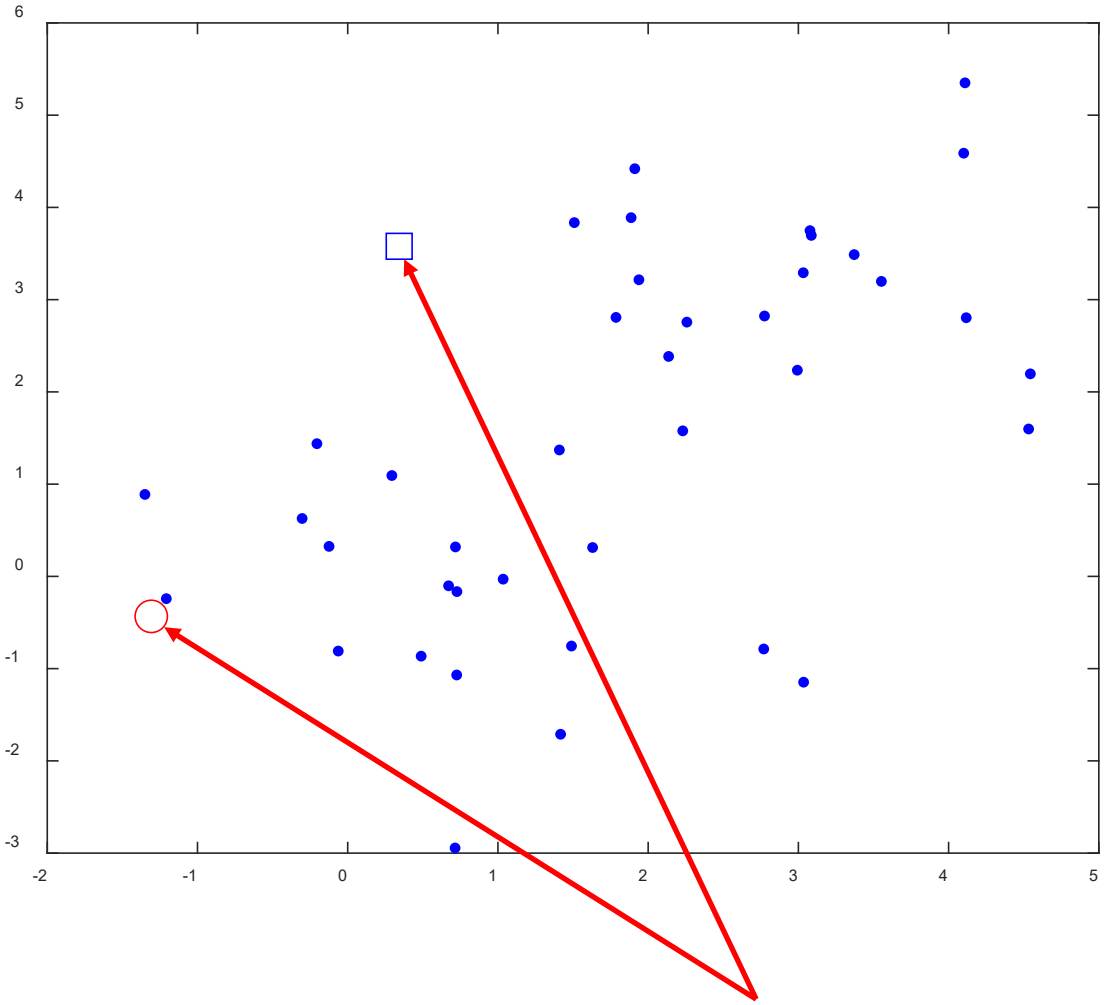
The k – mean clustering algorithm is summarized as follows:

Step-1: Set the number k to decide the number of clusters.

Step-2: Randomly generate k points or centroids as the initial cluster centres. They can also be randomly selected from the data.

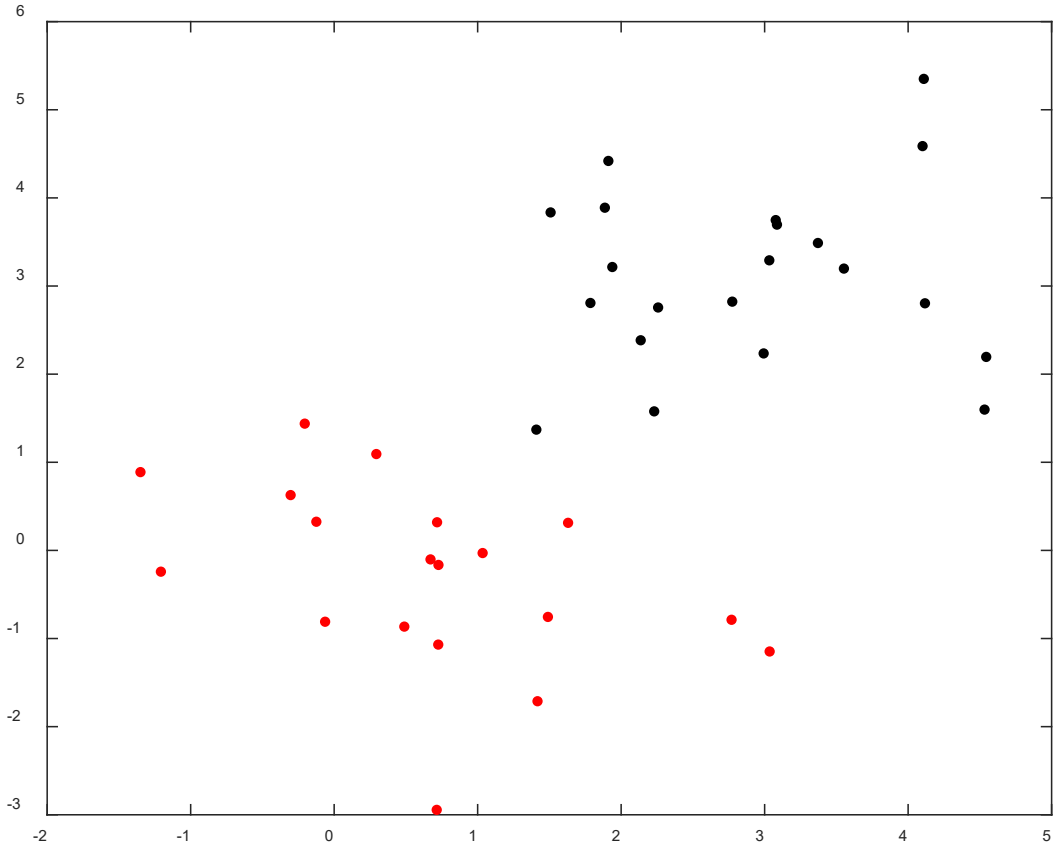


All samples

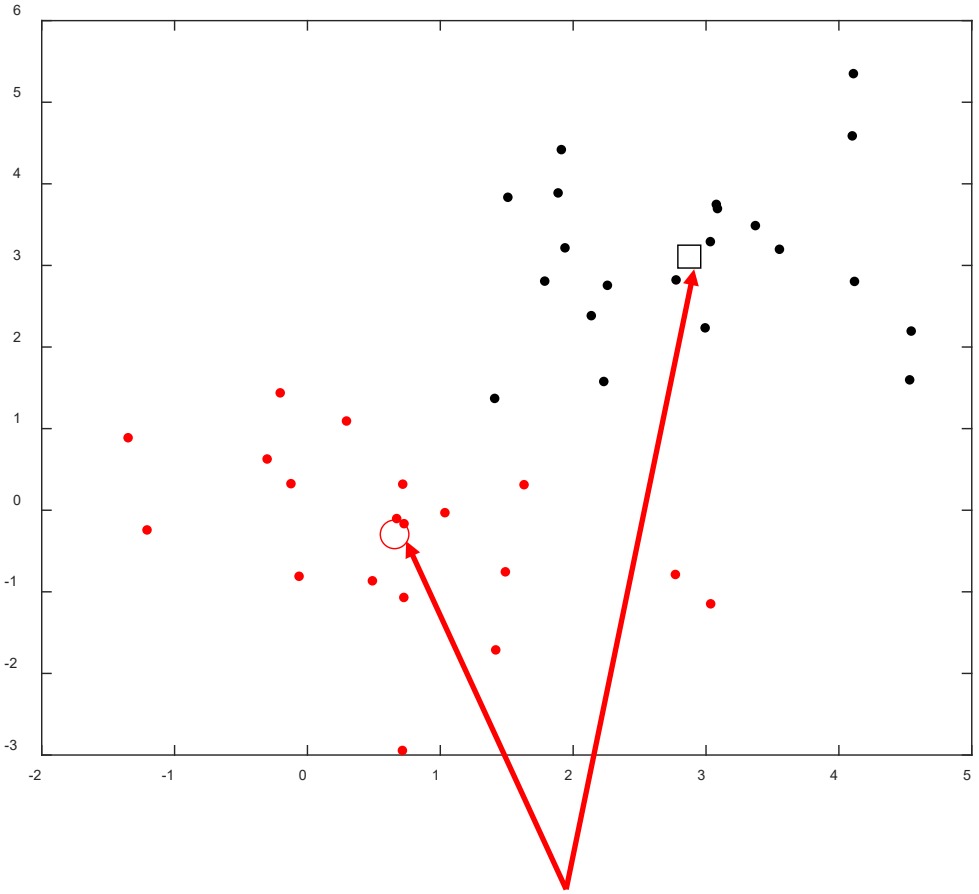


Step-2: Randomly generate k points as the initial cluster centres

Step-3: Assign each data point to their closest centroid, which will form the predefined k clusters.

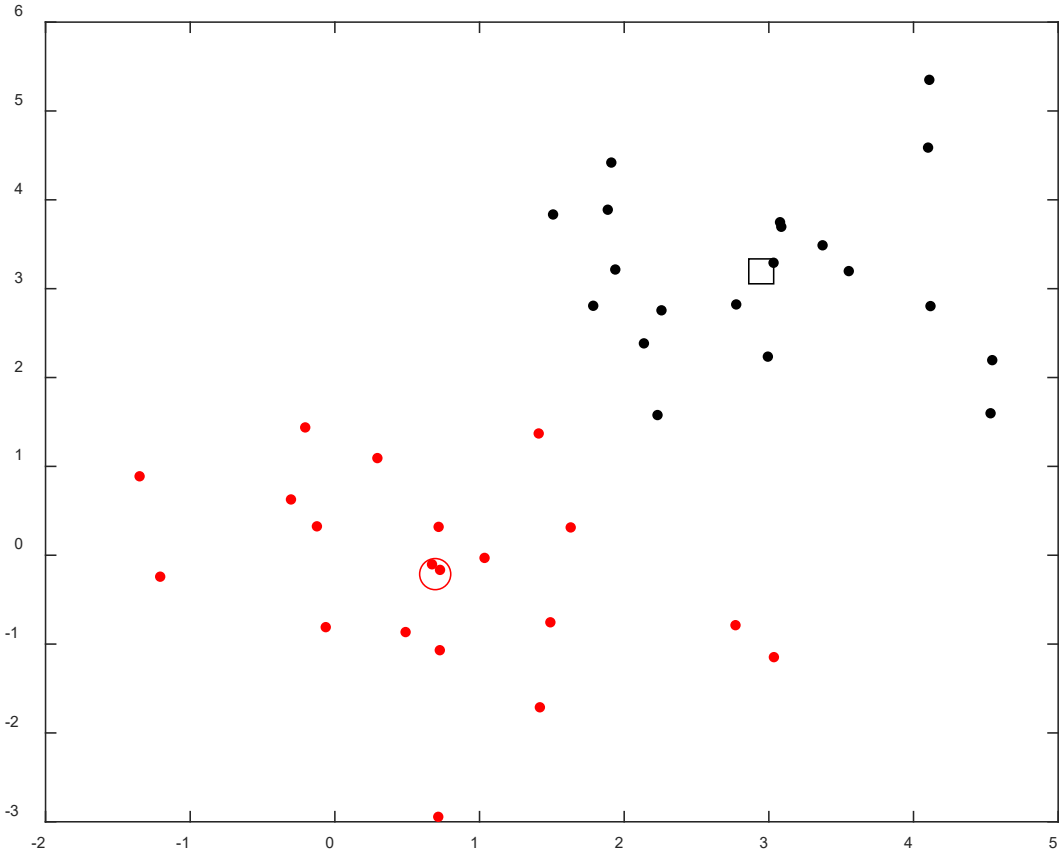


Step-4: Calculate a new centroid of each cluster.



new centroids

Step-5: Repeat Steps 3-4, until there is no change in the centroids and the assignment of samples

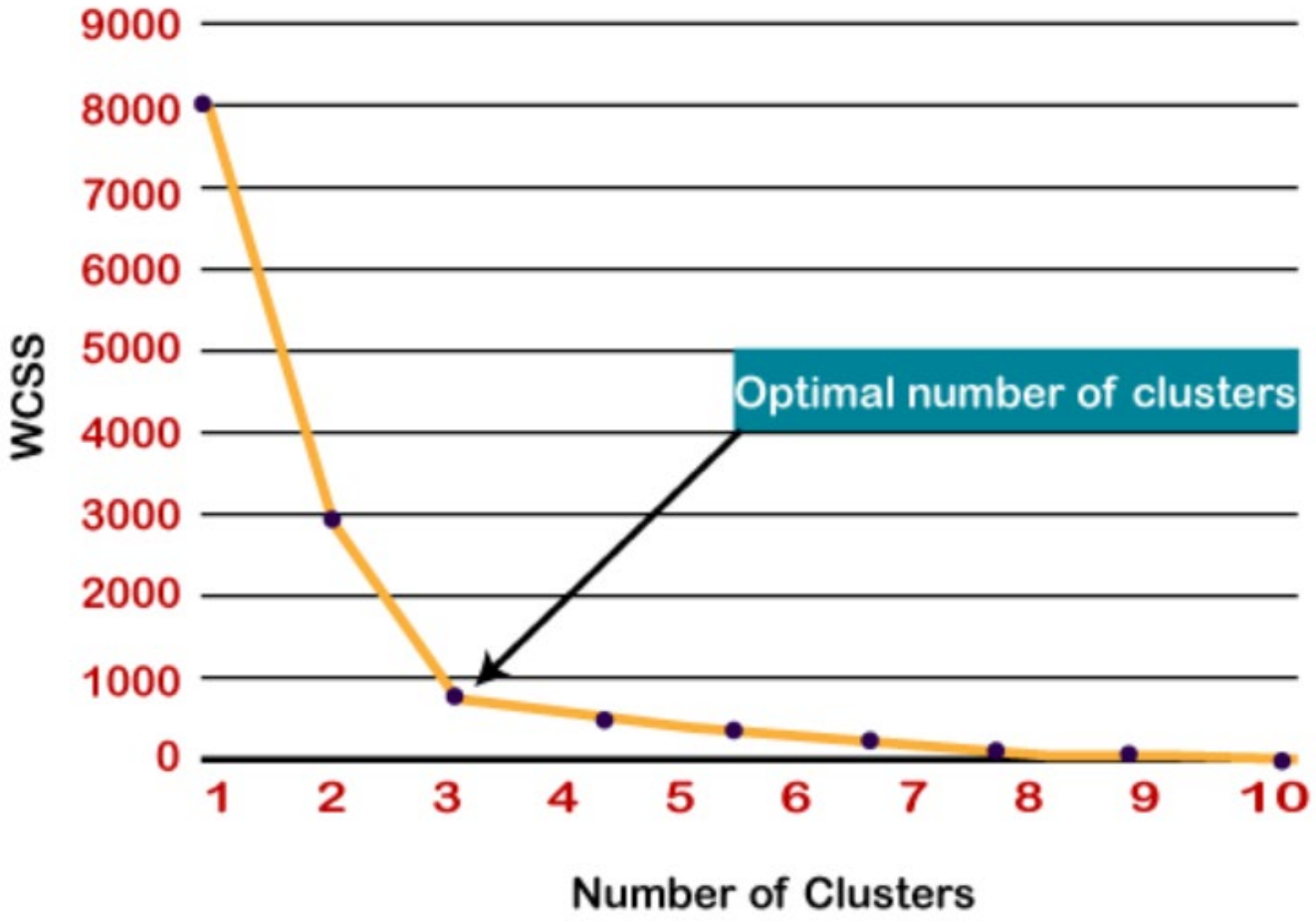


How to choose the value of k in k –means clustering?

The **Elbow method** is one of the most popular ways to find the optimal number of clusters. This method uses $WCSS$, which defines the total within cluster variations

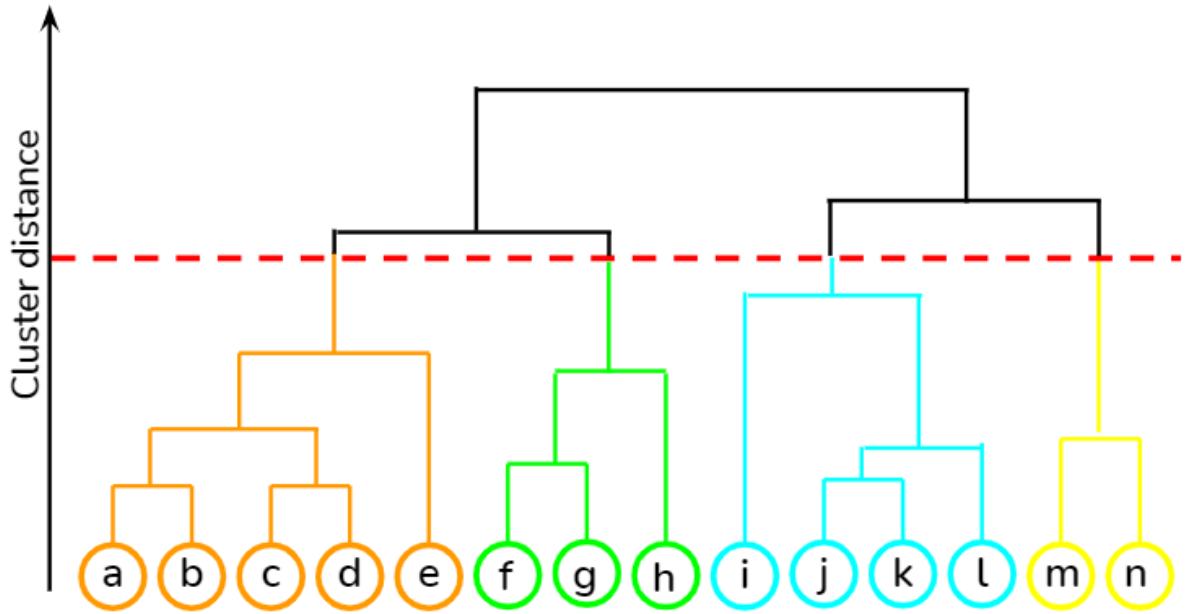
To find the optimal value of clusters, the elbow method follows the following steps:

(i) Execute the k –means clustering on a given dataset for different k values
(ii) For each value of k, calculates the $WCSS$ value.
(iii) Plots a curve of $WCSS$ values against k values
(iv) The sharp point of bend of the plot is considered as the best value of k.


9000
8000
7000
6000
5000
4000
3000
2000
1000
0
WCSS
Optimal number of clusters
1
2
3
4
5
6
7
8
9
10
Number of Clusters

10.4 Hierarchical Clustering

Hierarchical clustering is one of the most commonly used methods for summarizing the property of data structure. A hierarchical tree is a nested set of partition of samples, represented by a tree diagram or dendrogram as illustrated below:



There are several different algorithms for finding a hierarchical tree:

(1) An **agglomerative** algorithm, also known as the bottom-up approach, begins with N sub-clusters, each with one single sample and successively agglomerates pairs of clusters until all clusters have been merged into a single cluster that contains all samples

(2) A **divisive** algorithm, also known as the top-down approach, starts with a single clusters consisting of all samples, and successively splits the clusters, until each sub-cluster contains only one samples.

The advantage of hierarchical clustering is that it does not require us to specify the number of clusters, and this is different from the k-means clustering algorithm.

Agglomerative Algorithm

The agglomerative algorithm starts by calculating the similarity or dissimilarity between all the N samples, where each sample is considered as a cluster.

Consider the following example with 6 samples S1-S6, where each sample has 4 features A1, A2, A3, A4

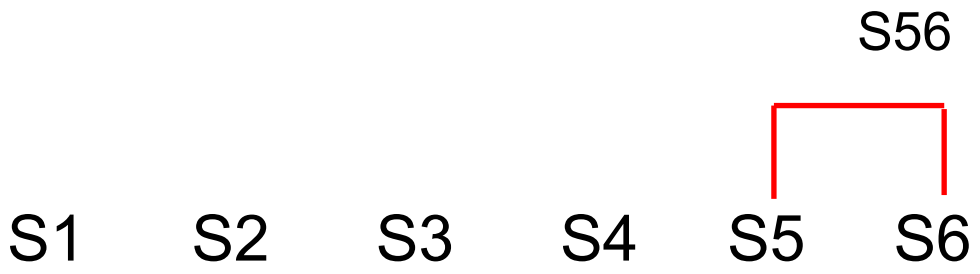
	A1	A2	A3	A4
S1	1.2426	0.783	-0.521	-0.00342
S2	0.5079	1.107	-1.212	2.48420
S3	0.0716	1.479	0.999	1.04288
S4	0.2323	0.231	-1.074	-0.18492
S5	0.2783	1.263	1.759	2.06782
S6	0.0257	0.399	0.861	1.86497

If the Euclidean distance is used as similarity measure, the similarity (proximity) matrix is obtained as follows:

	S1	S2	S3	S4	S5	S6
S1	0	2.704	2.294	1.290	3.263	2.651
S2	2.704	0	2.701	2.826	3.013	2.327
S3	2.294	2.701	0	2.718	1.311	1.365
S4	1.290	2.826	2.718	0	3.764	2.832
S5	3.263	3.013	1.311	3.764	0	1.288
S6	2.651	2.327	1.365	2.832	1.288	0

Then two clusters with the maximum similarity or minimum distance are clustered together.

By inspecting the distance matrix, we can see that S5 and S6 have the minimum distance 1.288, and hence are clustered together. We name the new cluster as S56 and draw the dendrogram as follows:



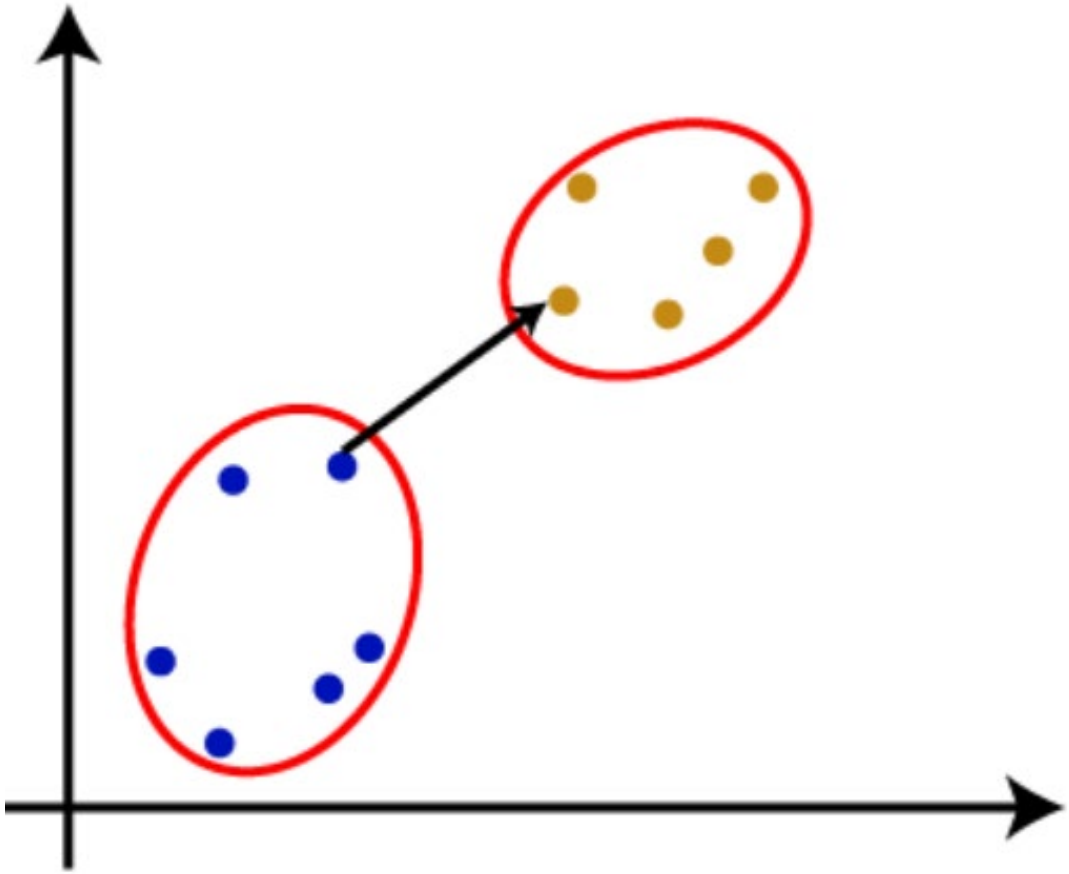
We next compute the similarity between the $N-1$ clusters. S56 is a newly merged clusters, there are a few ways to calculate its similarity to other clusters, such as

(1) Single linkage (nearest neighbour)

In single-linkage, the distance between two clusters is defined as the minimum value of all pairwise distances between the samples in one cluster and the samples in another cluster.

For example, the distance between S3 and S56 is obtained as:

$$d_{S3,S56} = \min\{d_{S3,S6}, d_{S3,S5}\}$$
$$= \min\{1.365, 1.311\} = 1.311$$



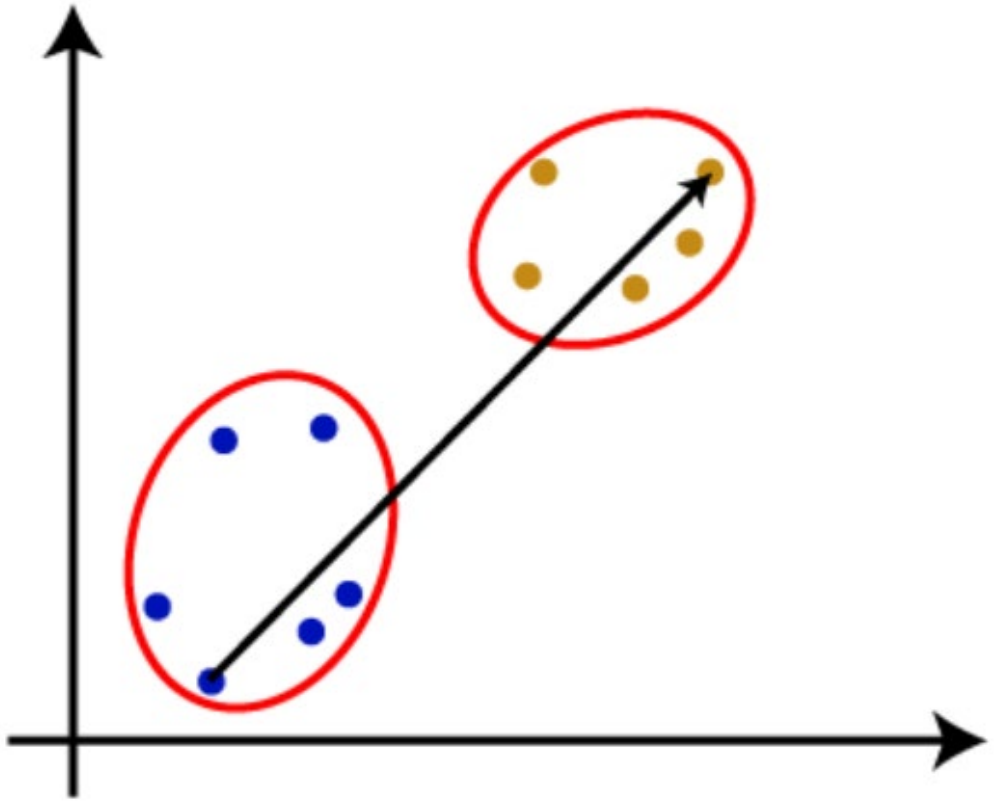
Single linkage

(2) Complete linkage (farthest neighbour)

In complete linkage, the distance between two clusters is defined as the maximum value (farthest) of all pairwise distances between the samples in one cluster and the samples in another cluster.

For example, the distance between S3 and S56 is obtained as:

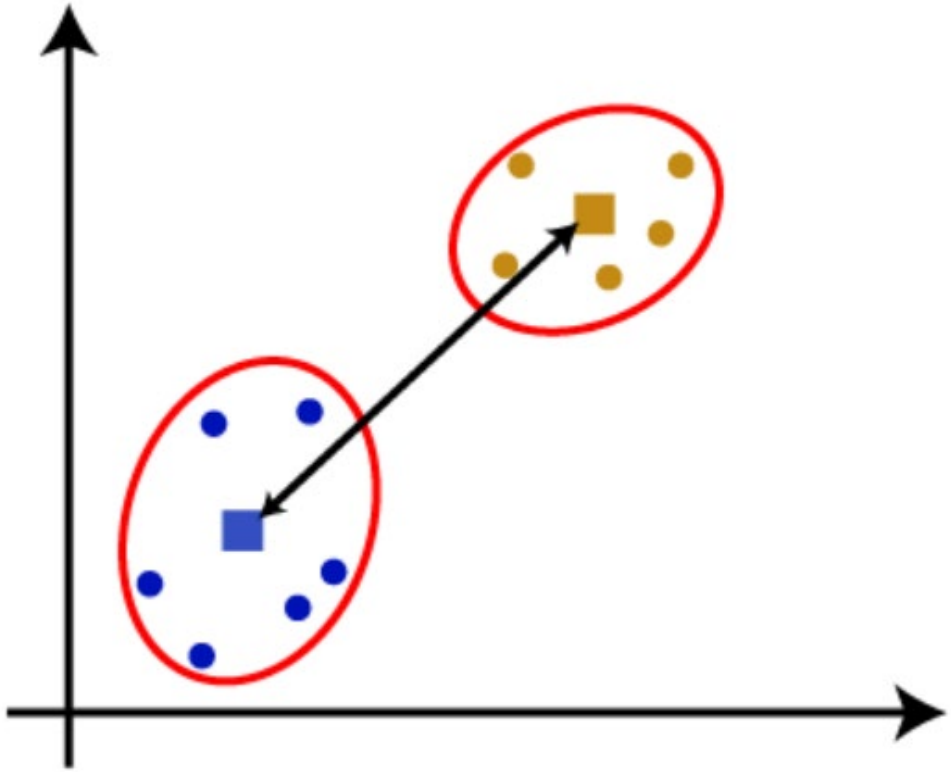
$$d_{S3,S56} = \max\{d_{S3,S6}, d_{S3,S5}\}$$
$$= \max\{1.365, 1.311\} = 1.365$$



Complete linkage

(3) Centroid linkage

In the centroid linkage, the distance between two clusters is defined as the distance between the two centroids of the two clusters.



In the above example, the centroid of cluster S56 is:

C_{S56}=[0.1520 0.8310 1.3100 1.9664]

The centroid of cluster S3 remains unchanged, then:

$$d_{S3,S56} = \sqrt{(C_{S56} - S3) * (C_{S56} - S3)'}$$

$$=1.173$$

(4) Average Linkage

It is the linkage method in which the distance between each pair of data is added up and then divided by the total number of data pairs to calculate the average distance between two clusters. It is also one of the most popular linkage methods.

In the above example, from the distance matrix, we have:

$$d_{S3,S5} = 1.311$$

$$d_{S3,S6} = 1.365$$

Then:

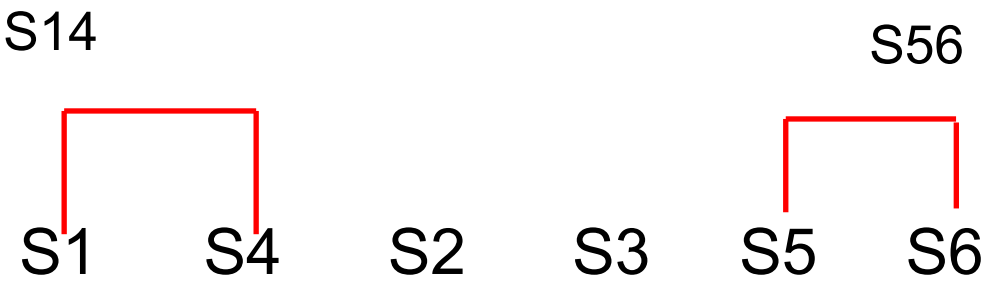
$$d_{S3,S56} = \text{Average}\{1.311, 1.365\} = 1.338$$

If single linkage is used, the similarity matrix is as follows:

	S1	S2	S3	S4	S56
S1	0	2.704	2.294	1.290	2.651
S2	2.704	0	2.701	2.826	2.327
S3	2.294	2.701	0	2.718	1.311
S4	1.290	2.826	2.718	0	2.832
S56	2.651	2.327	1.311	2.832	0

Obviously, S1 and S4 have the minimum distance 1.290 or the maximum similarity and are clustered together. The new cluster is named as S14.

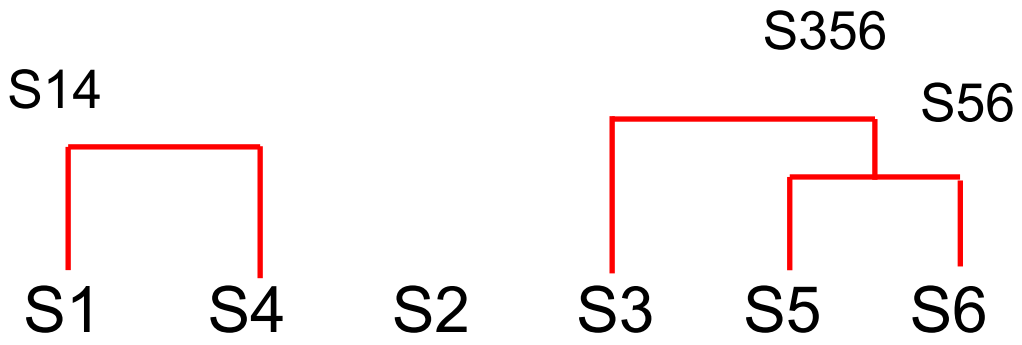
The dendrogram is as follow:



Next, we computer the similarity matrix of the $N - 2$ clusters.

	S14	S2	S3	S56
S14	0	2.704	2.294	2.651
S2	2.704	0	2.701	2.327
S3	2.294	2.701	0	1.311
S56	2.651	2.327	1.311	0

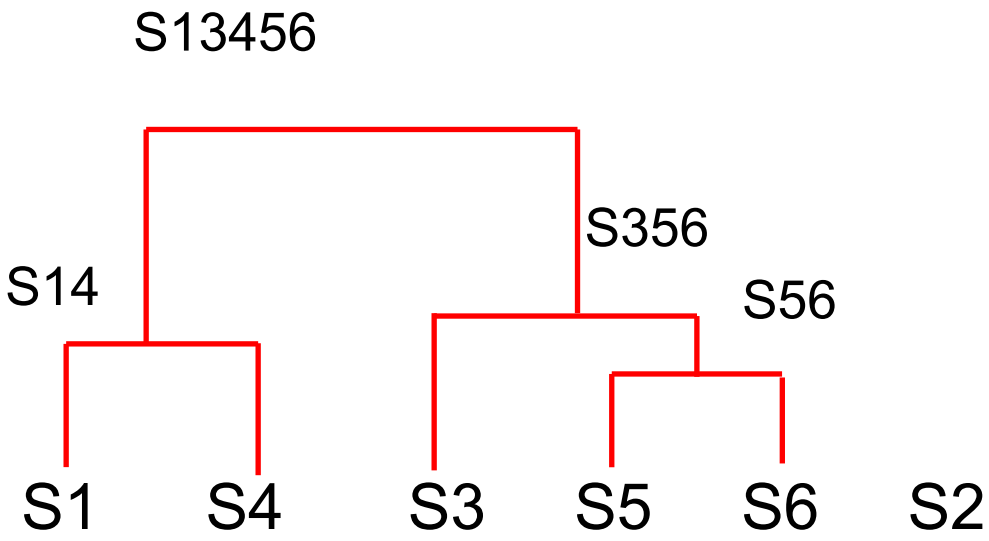
Obviously, S3 and S56 have the minimum distance 1.311 or the maximum similarity and are clustered together. The new cluster is named as S356


S14
S356
S56
S1
S4
S2
S3
S5
S6

Next, we compute the similarity matrix of the $N - 3$ clusters.

	S14	S2	S356
S14	0	2.704	2.294
S2	2.704	0	2.327
S356	2.294	2.327	0

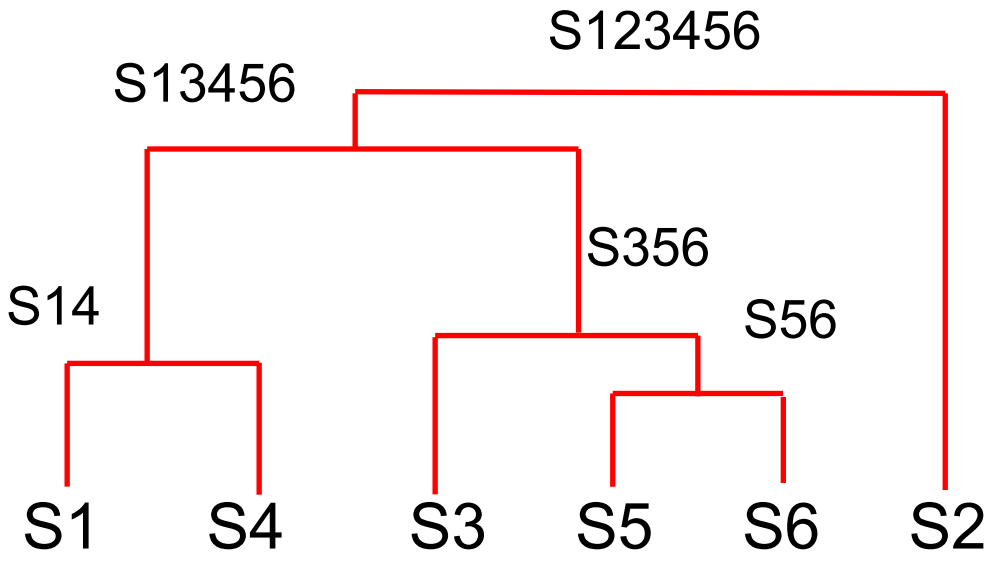
Obviously, S14 and S356 have the minimum distance 2.294, and are therefore clustered together. We name the new cluster as S13456 as shown next:


S13456
S356
S14
S56
S1
S4
S3
S5
S6
S2

We now just have 2 clusters

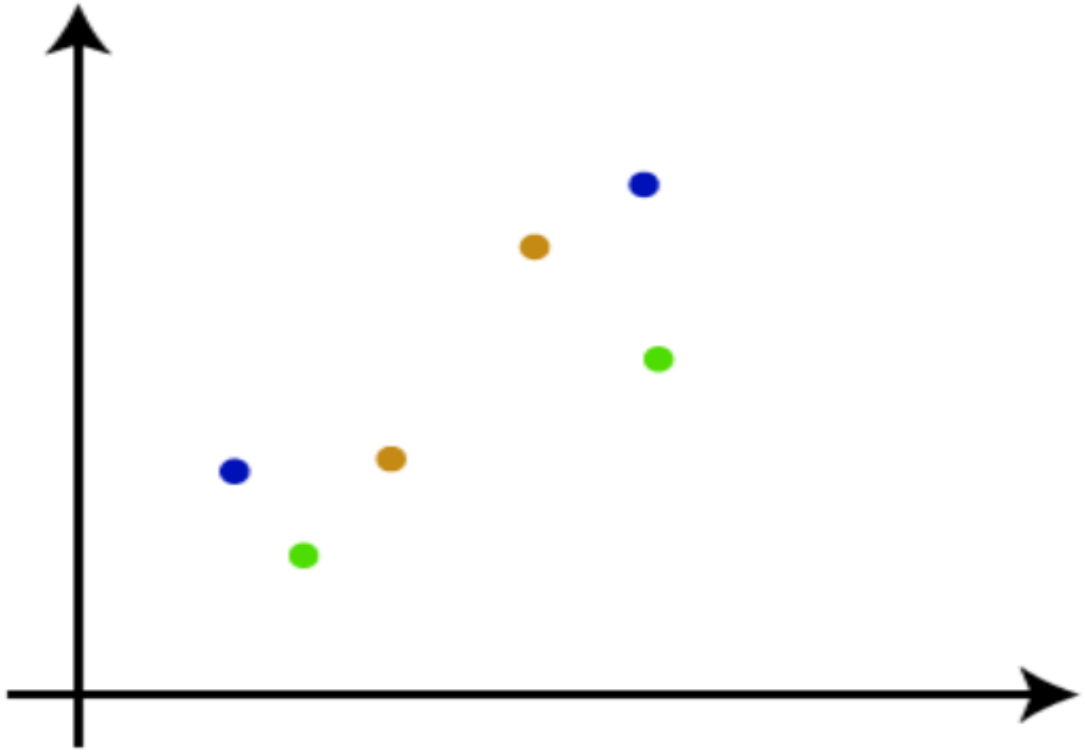
	S13456	S2
S13456	0	2.327
S2	2.327	0

Then the final dendrogram is as follow:

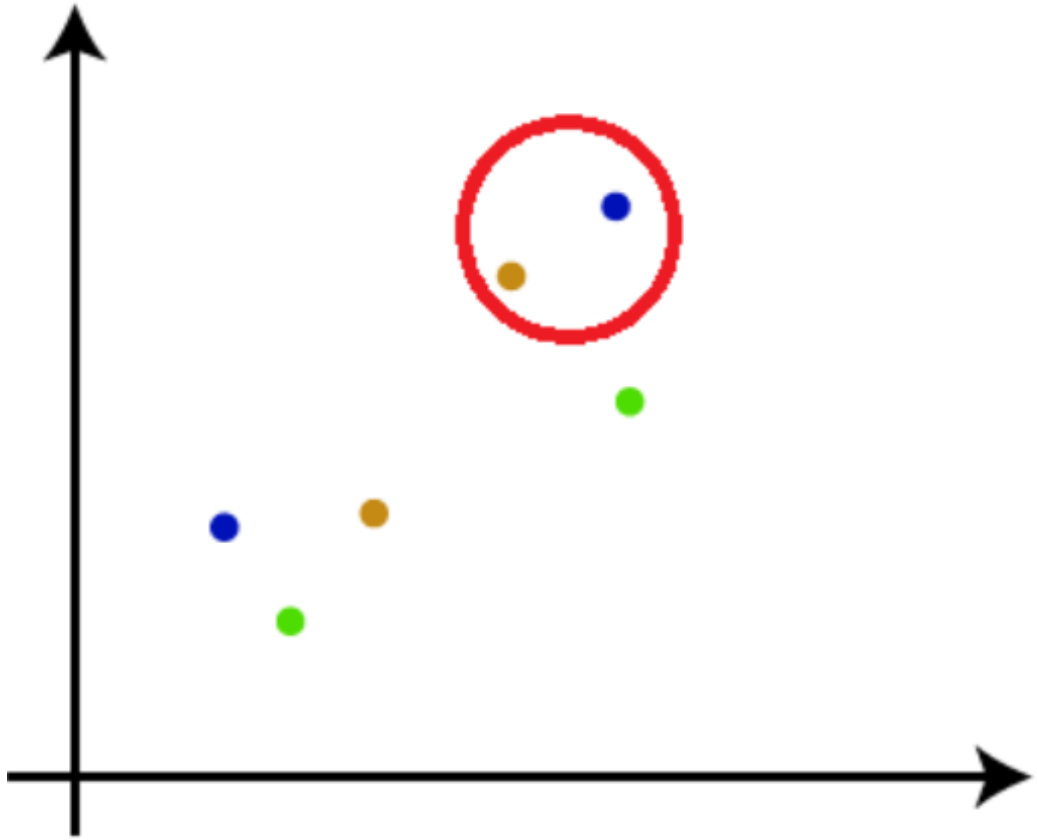


Summary of Agglomerative Hierarchical Clustering Algorithm

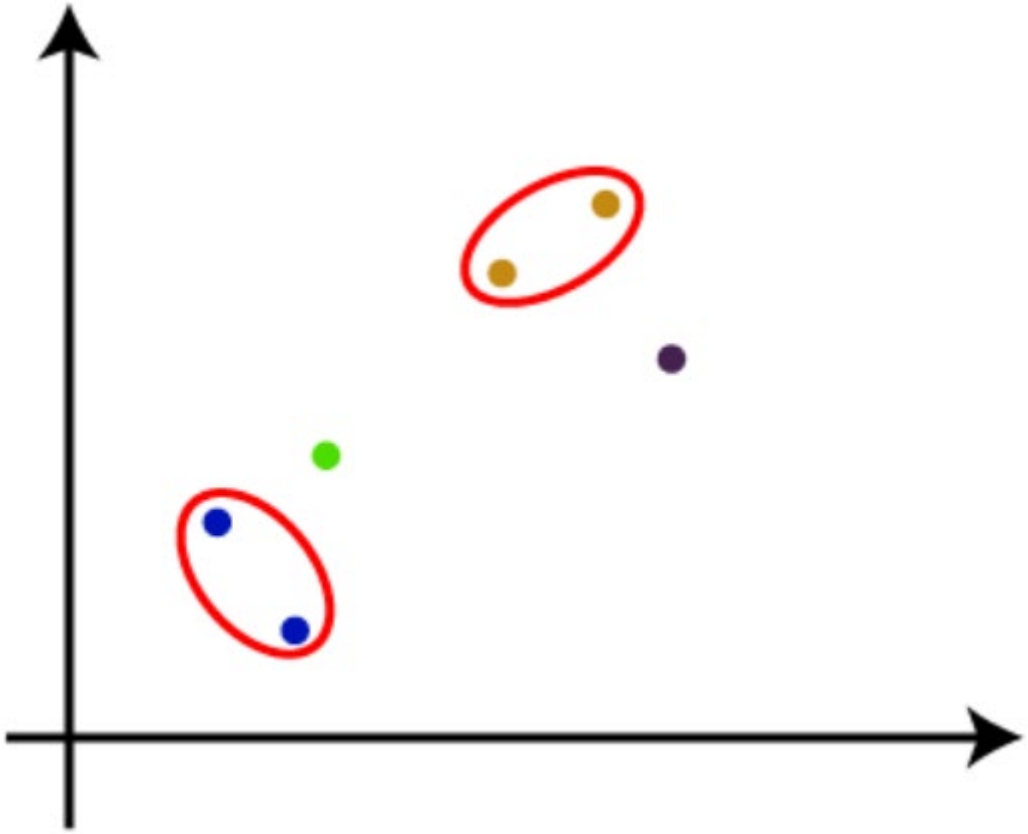
Step-1: Consider each sample (data point) as a single cluster. If there are N data points, the number of clusters will be N



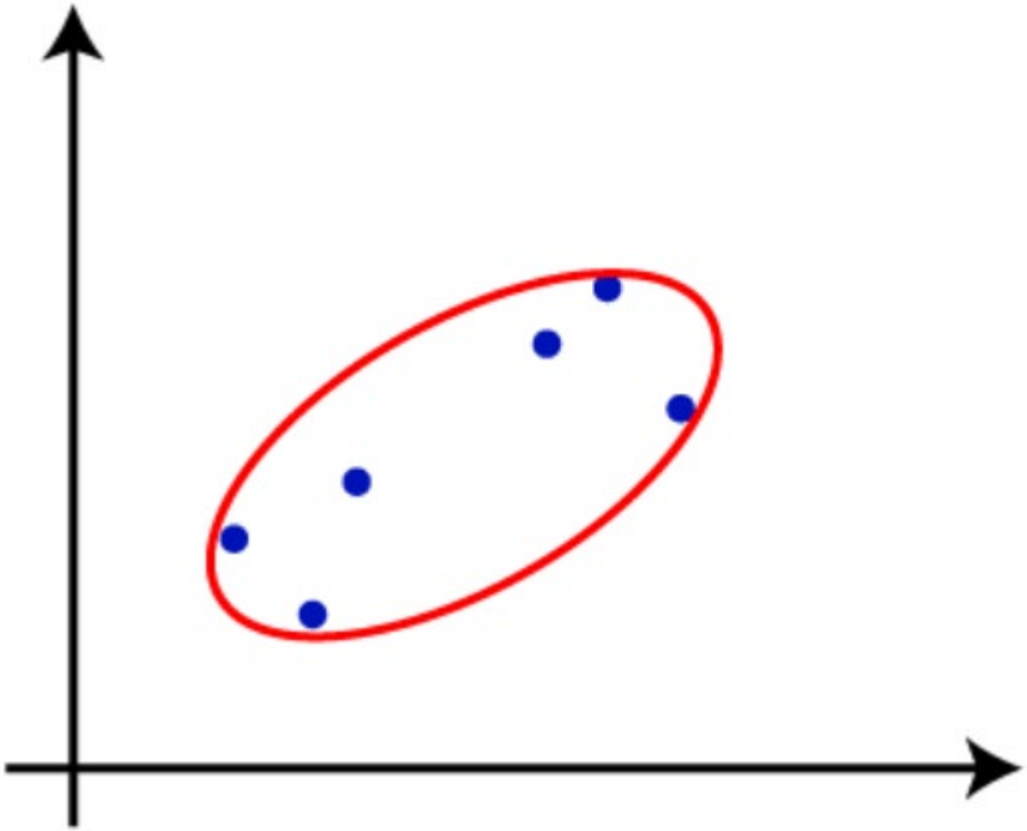
Step-2: Take two closest data points or clusters and merge them to form one cluster. So, there will now be $N-1$ clusters.



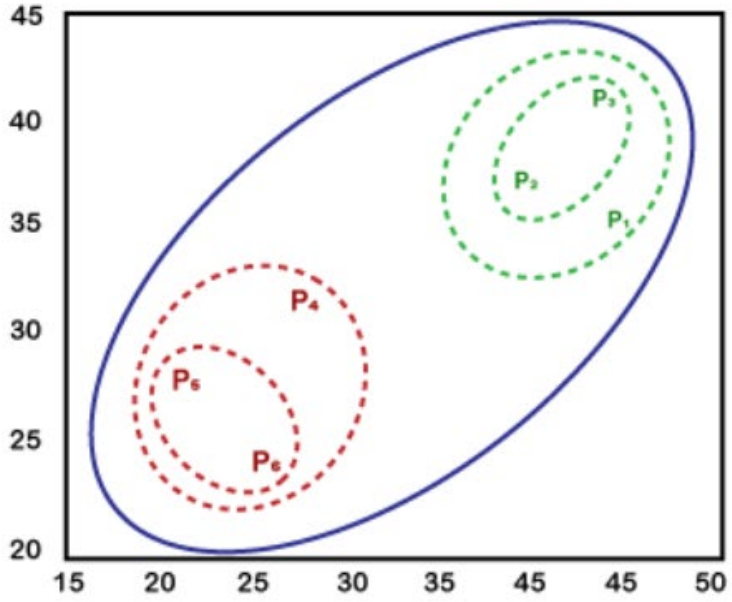
Step-3: Again, take the two closest clusters and merge them together to form one cluster. There will be $N - 2$ clusters.

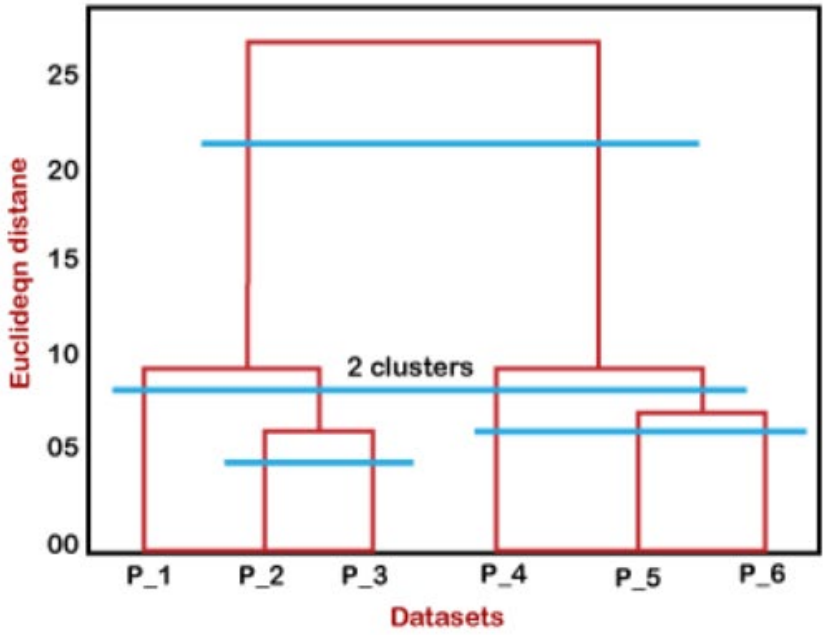


Step-4: Repeat Step-3 until all samples are merged into one cluster as shown below:



Step-5: Once all the clusters are combined into one big cluster, we develop the dendrogram to divide the clusters as per the problem.





In the above diagram, the left part is showing how clusters are created in agglomerative clustering, and the right part is showing the corresponding dendrogram.

(i) Firstly, the datapoints P2 and P3 combine together and form a cluster, correspondingly a dendrogram is created. The height is decided according to the Euclidean distance between the data points.
(ii) Secondly, P5 and P6 form a cluster, and the corresponding dendrogram is created. It is higher than the previous, as the Euclidean distance between P5 and P6 is a little bit greater than the P2 and P3.
(iii) Next, P1, P2, and P3 ae combined in one dendrogram, and P4, P5, and P6 in another dendrogram.
(iv) At last, the final dendrogram is created that combines all the data points together.

10.5 Distribution-based Clustering

In real life, many datasets can be modeled by Gaussian distribution (univariate or multivariate). It is therefore quite natural and intuitive to assume that the data comes from different Gaussian distributions. Hence, we can try to model the dataset as a mixture of several Gaussian distributions. This is the core idea of the Gaussian Mixture Model (GMM).

GMM has been used to model class-conditional probability density function in supervised learning. Here it is used for clustering in unsupervised learning.

Assume there are m clusters underlying the data, and the data in each cluster follows normal (Gaussian) distribution. Then the distribution of the data can be modelled as :

$$p(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i N(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

Where $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the mean vector (i.e. cluster centre) and covariance matrix of the i^{th} Gaussian component (i.e. cluster), respectively. α_i is the weight of the i^{th} Gaussian component, and

$$\sum_{i=1}^{m} \alpha_i = 1$$

The GMM model parameter can be estimated using the Expectation-Maximization (EM) algorithm. Please refer to lecture notes of Part 3 for details of the EM-based parameter estimation of the GMM model.

Procedure of the distribution-based clustering algorithm

Step-1: Set the number m to decide the number of clusters.

Step-2: initialize $\hat{\alpha}_i(0), \hat{\boldsymbol{\mu}}_i(0), \hat{\boldsymbol{\Sigma}}_i(0)$ with random values.

Step-3: Estimate the parameters of Gaussian mixture model using the Expectation-Maximization algorithm.

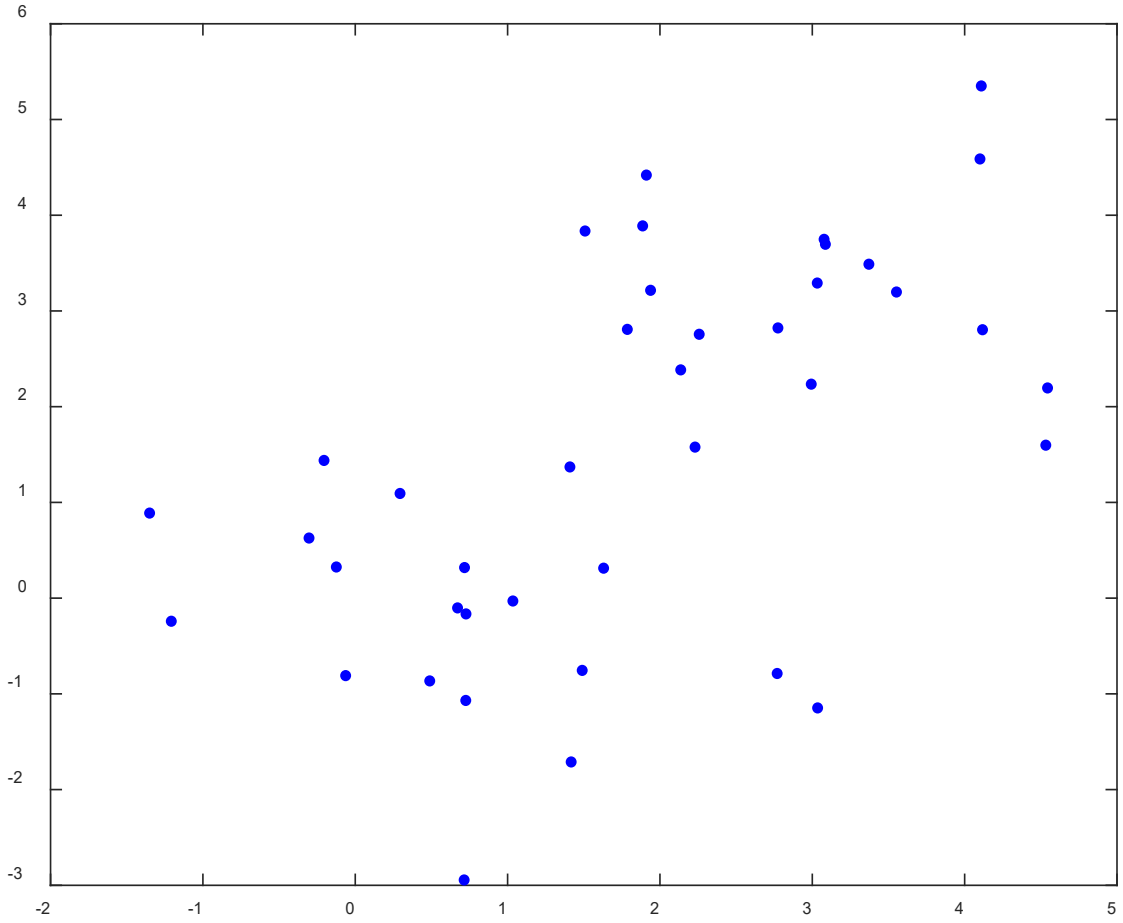
Step-4: The mean vectors of the Gaussian components are the cluster centres. Assign each data point to their closest cluster centre.

In Matlab, we can use the function *fitgmdist* to fit a GMM to the data.

$$gmm = fitgmdst(data, m);$$

Where m is the number of clusters.

Example: There are 40 samples as shown below:



Assume there are two clusters:

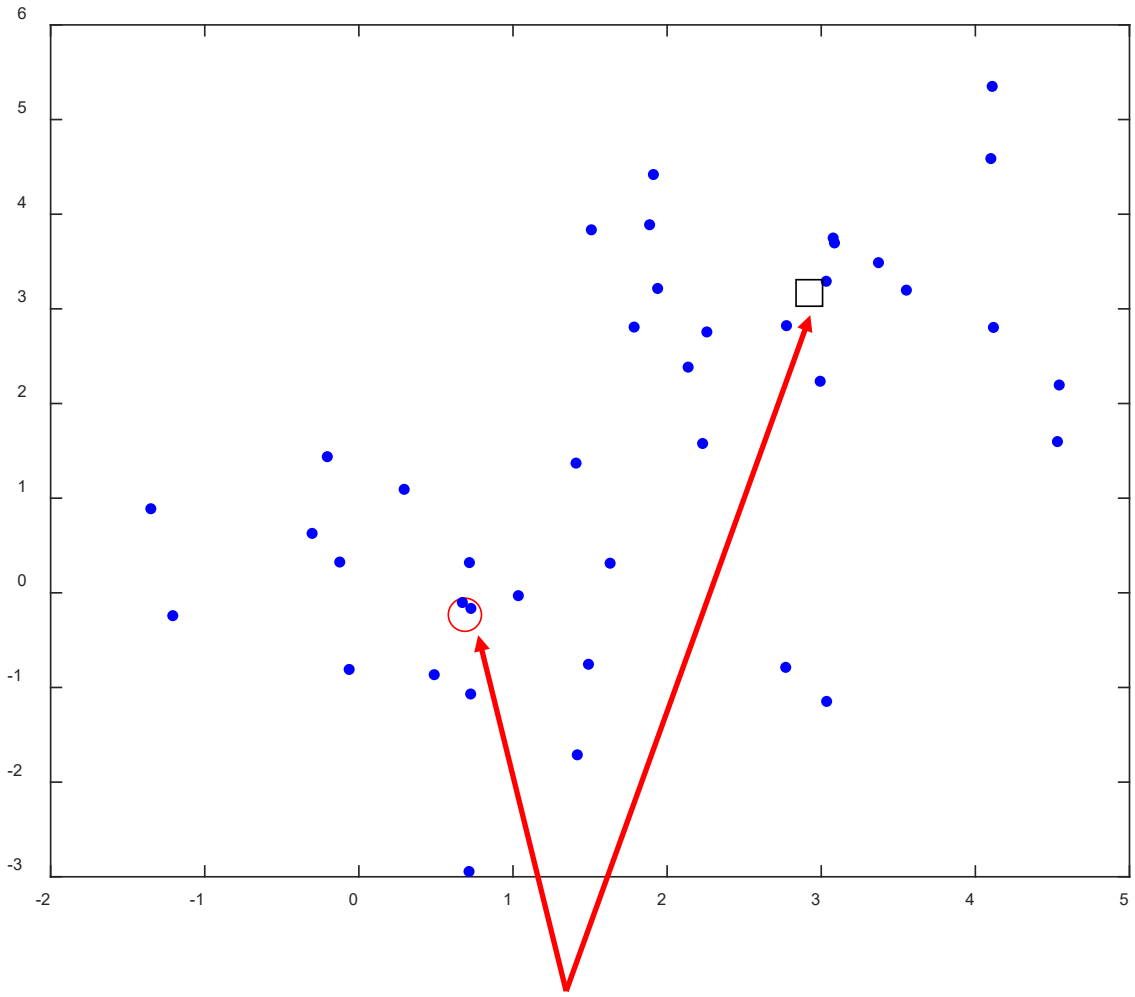
$$gmm = fitgmdst(data, 2);$$

The result is:

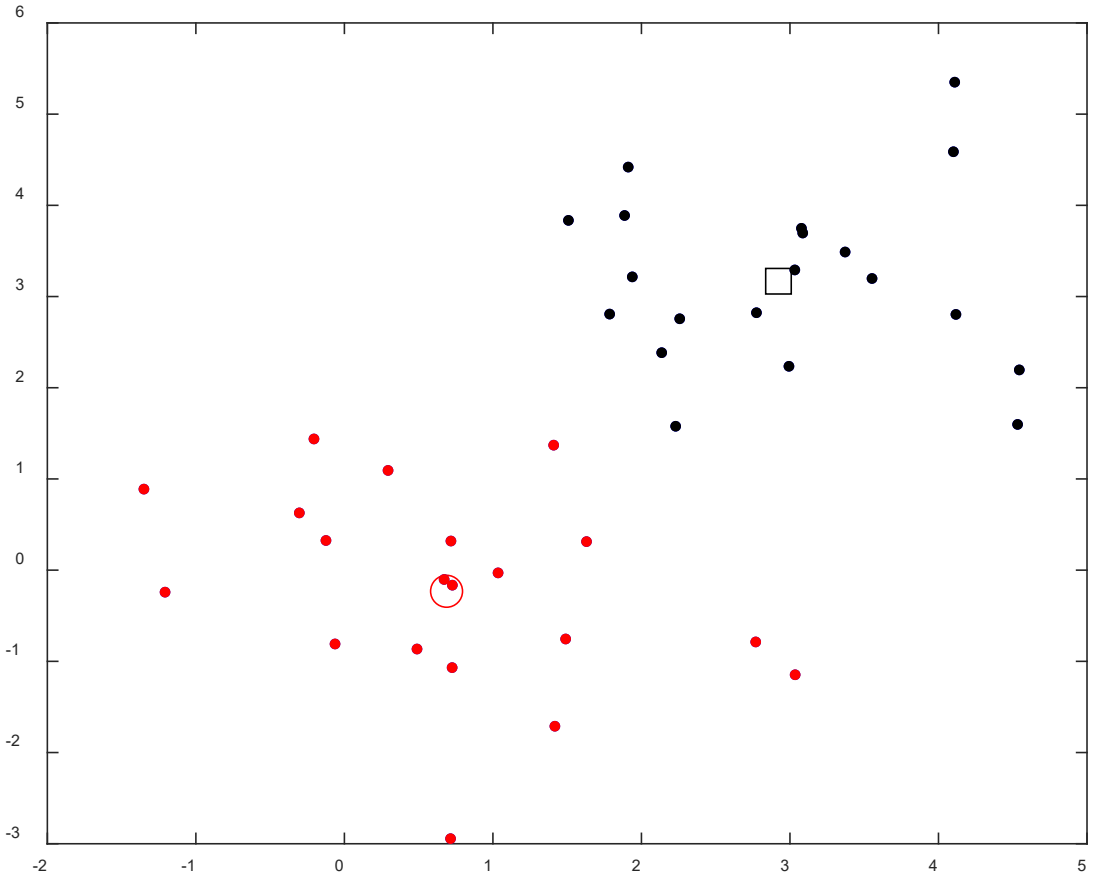
```
gmm =

Gaussian mixture distribution with 2 components in 2 dimensions
Component 1:
Mixing proportion: 0.493218
Mean:    0.6879   -0.2316

Component 2:
Mixing proportion: 0.506782
Mean:    2.9226    3.1679
```



Cluster Centres



Each sample is assigned to its closest cluster centre

If we assume there are three clusters:

$$gmm = fitgmdst(data, 3);$$

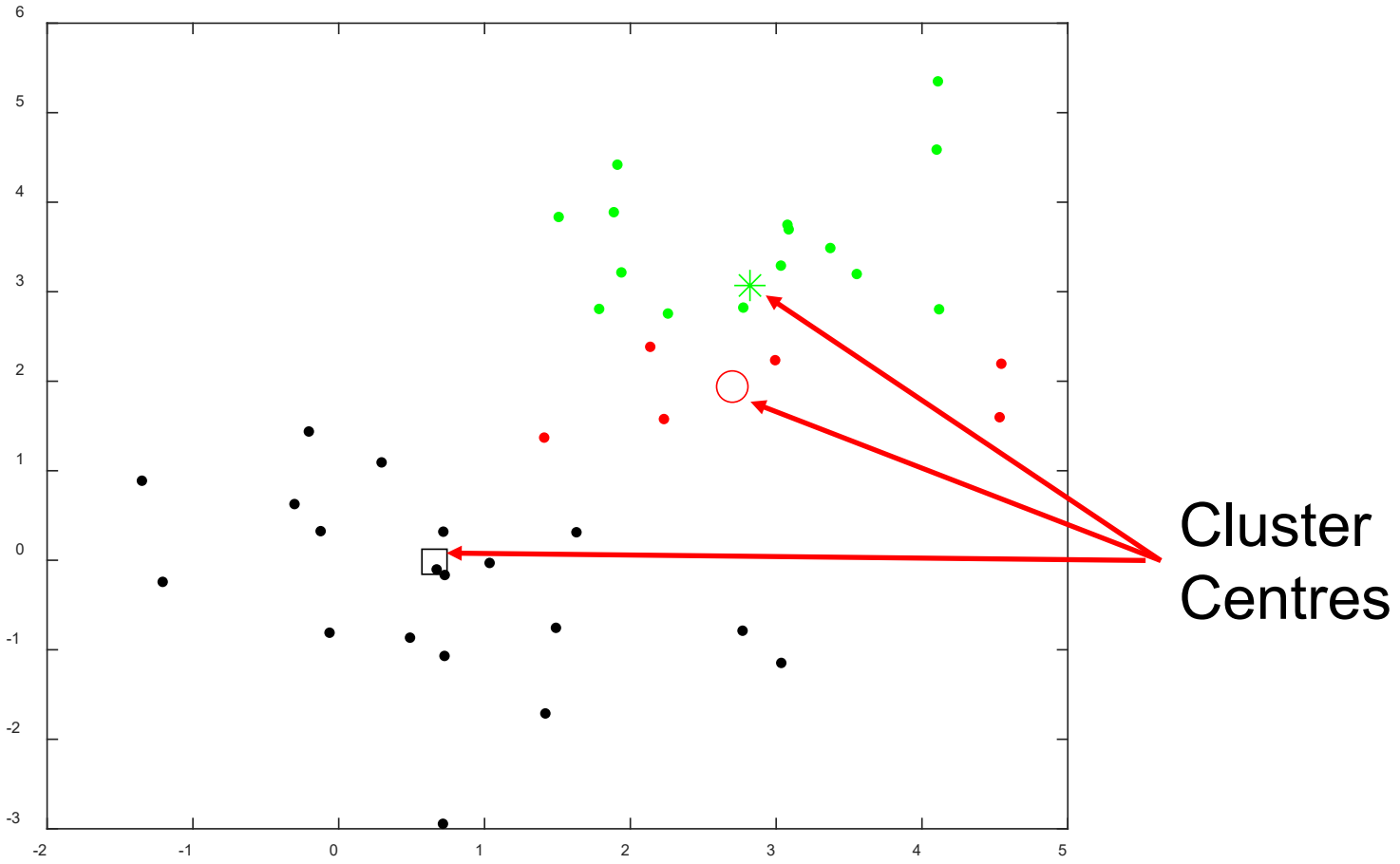
The result is:

```
gmm =

Gaussian mixture distribution with 3 components in 2 dimensions
Component 1:
Mixing proportion: 0.159050
Mean:    2.6990    1.9401

Component 2:
Mixing proportion: 0.453197
Mean:    0.6566   -0.0167

Component 3:
Mixing proportion: 0.387753
Mean:    2.8203    3.0695
```



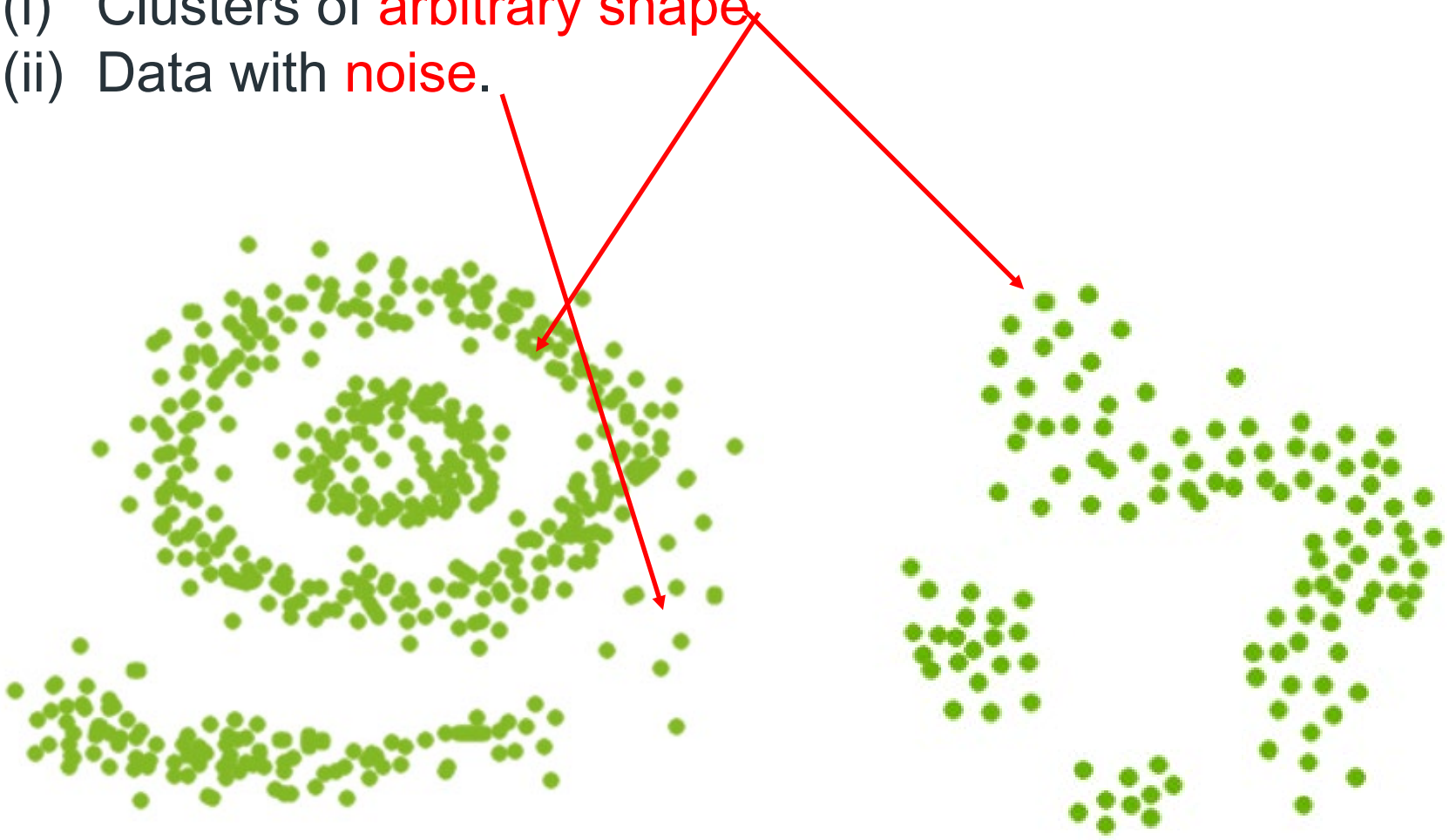
Only 6 samples belong to cluster 1. The suitable number of clusters can be obtained by setting a threshold on the number of samples in a cluster.

10.6 Density-based Clustering

K-means and hierarchical clustering work well for finding spherical-shaped clusters or convex clusters. In other words, they are suitable only for compact and well-separated clusters as shown below:

But many real datasets may contain irregularities, such as:

(i) Clusters of arbitrary shape,
(ii) Data with noise.



Density-based clustering could solve the problems. Density-based spatial clustering of applications with noise (DBSCAN) clustering method is the most popular density-based clustering method.

The *DBSCAN algorithm* is based on the intuitive notion of "clusters" and "noise".

Clusters are dense regions in the data space, separated by regions of the lower density of points. For each point of a cluster, the neighbourhood of a given radius has to contain at least a minimum number of points.

By contrast, points in the lower density regions are considered as noise. This is very different from k-means clustering, where every data point is assigned to a cluster.

The DBSCAN algorithm uses two parameters:

(i) MinPts: The minimum number of points (a threshold) clustered together for a region to be considered dense. For example, MinPts = 4 means minimum 4 points are required to form a dense cluster

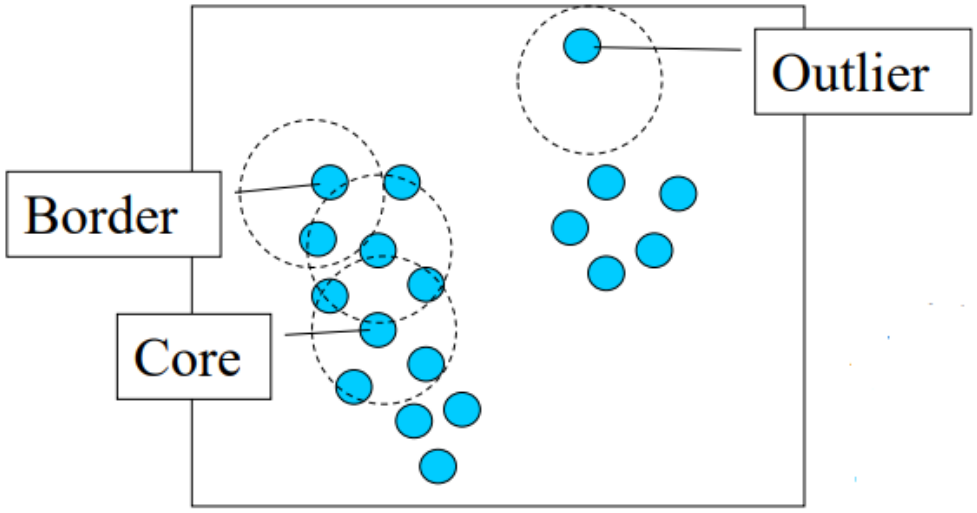
(ii) eps (ε): A distance measure that will be used to locate the points in the neighborhood of any point, i.e. if the distance between two points is lower or equal to 'eps' then they are considered as neighbors, i.e. ε-neighborhood.

In this algorithm, we have 3 types of data points:

(i) Core point: A point is a core point if it has more than $MinPts$ points within ε.

(ii) Border point: A point which has fewer than $MinPts$ within ε but it is in the neighborhood of a core point.

(iii) Noise or outlier: A point which is not a core point or border point



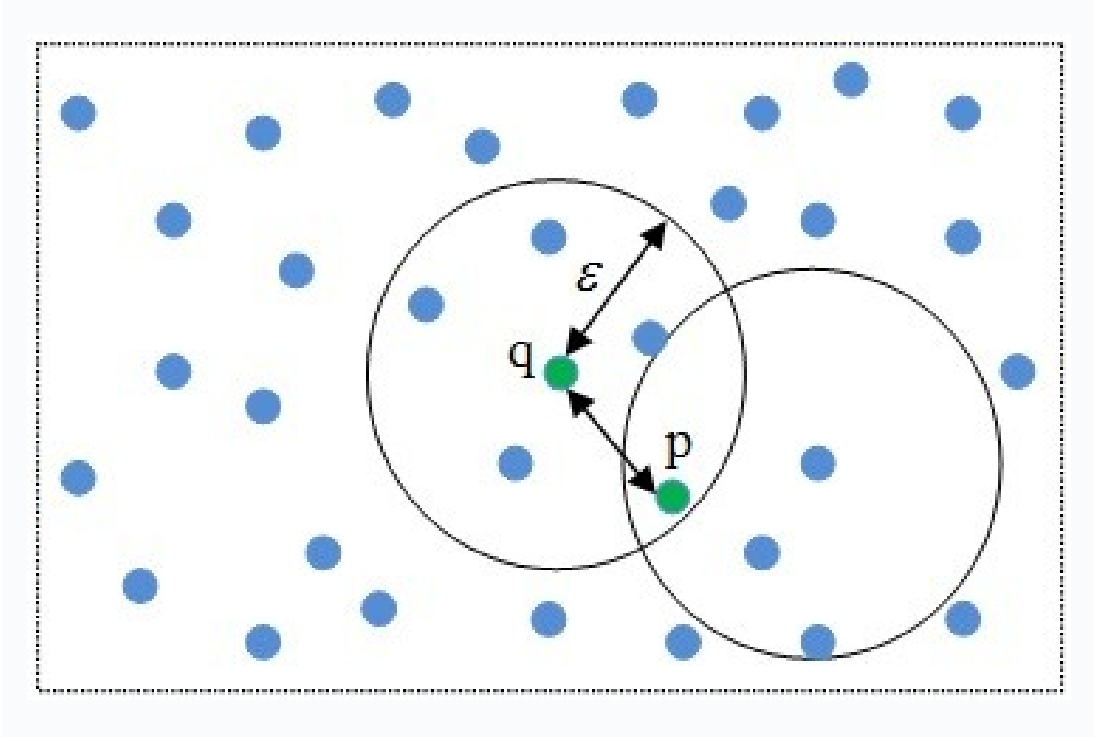
ε = 1unit, MinPts = 5

These parameters can be better understood if we explore two concepts called Reachability and Connectivity.

Reachability states if a data point can be accessed from another data point directly or indirectly, whereas Connectivity states whether two data points belong to the same cluster or not. In terms of reachability and connectivity, two points in DBSCAN may be:

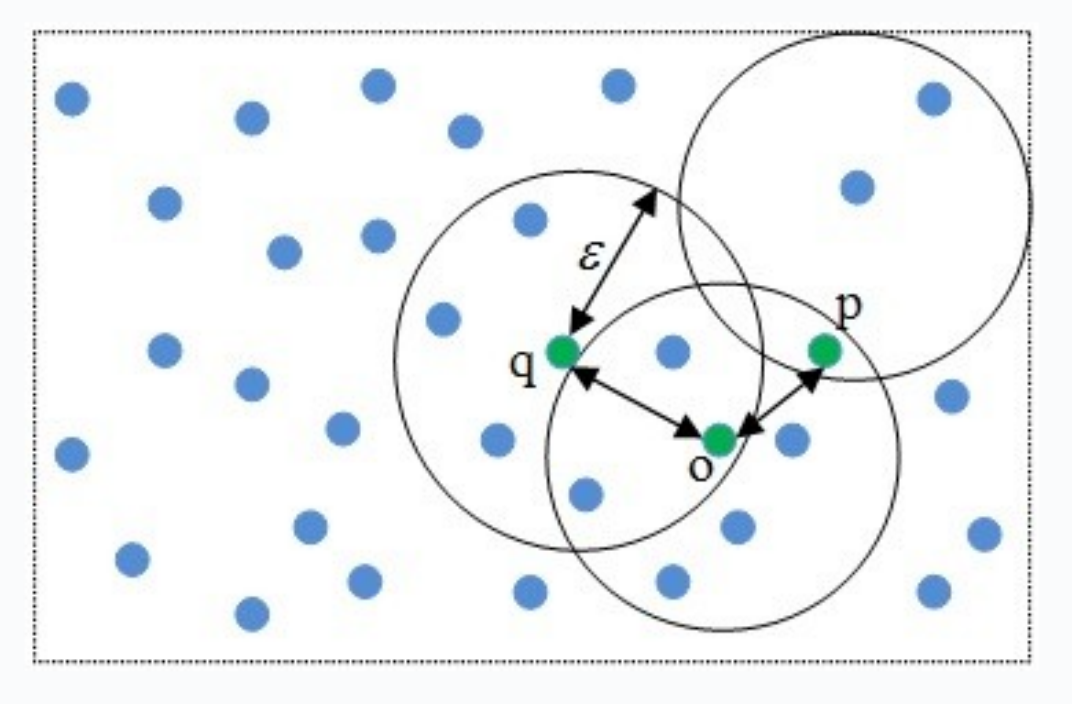
(i) Directly Density-Reachable
(ii) Density-Reachable
(iii) Density-Connected

A point (or sample) p is directly density reachable from point q if p is within the ε-Neighborhood of core point q



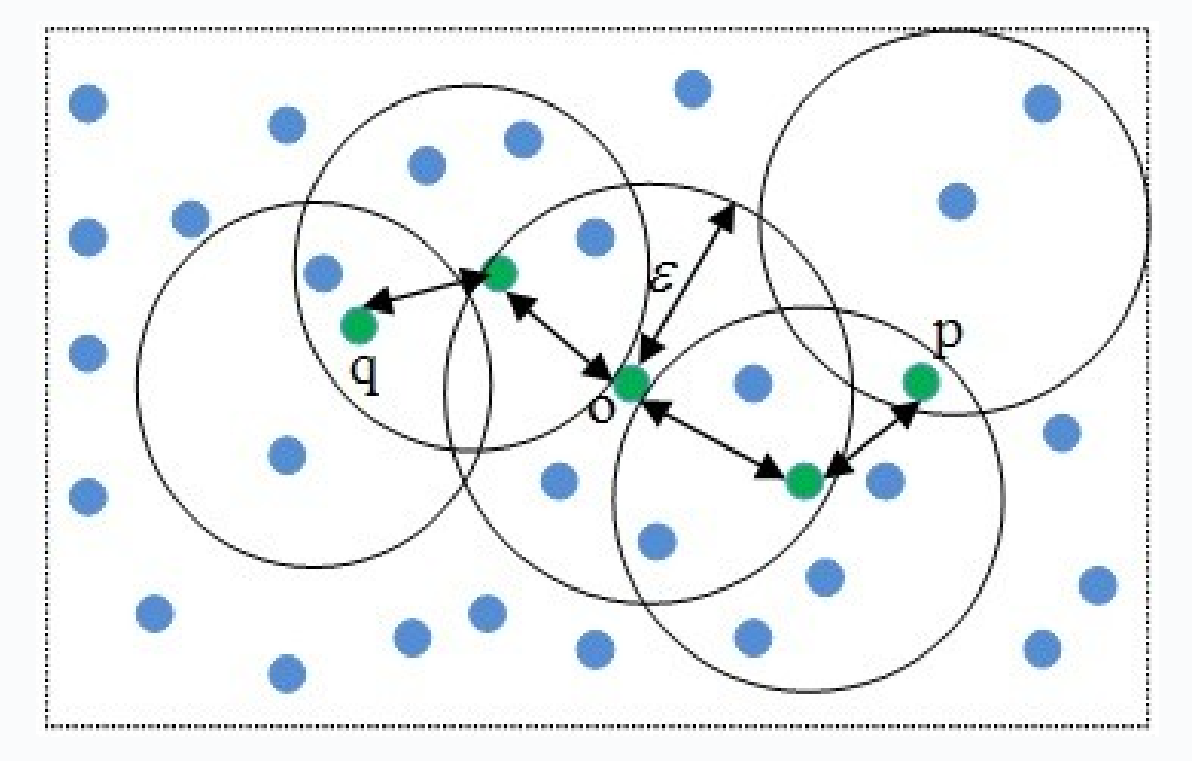
$$MinPts = \mathbf{5}$$

p is density-reachable from q if there are a sets of core points leading from q to p.



o is directly reachable from core point q, p is directly reachable from core point o. Hence p is reachable from q

Two points are called density connected if there is a core point that both points are density-reachable from the core point.



Points p and q are density-connected by core point o.

Parameter setting for DBSCAN

The DBSCAN has two parameters, including ε and $MinPts$.

(i) $MinPts$

As a rule of thumb, a minimum $MinPts$ can be derived from the number of dimensions D in the data set, such as $MinPts \geq D + 1$. The minimum value of $MinPts$ must be chosen at least 3

:

However, larger values are usually better for data sets with noise and will yield more significant clusters. As a rule of thumb, $MinPts = 2 \cdot D$ can be used, but it may be necessary to choose larger values for very large data, for noisy data or for data that contains many duplicates.

(ii) ε

The value for ε can then be chosen by plotting the distance to the $k = MinPts - 1$ nearest neighbor ordered from the largest to the smallest value. Good values of ε are where this plot shows an "elbow" (maximum curvature). In general, small values of ε are preferable, and as a rule of thumb, only a small fraction of points should be within this distance of each other.

DBSCAN searches for clusters in a dataset by checking ε-neighborhood of each point in the dataset. If the ε-neighborhood of a data point contains more than a minimum number of neighbors, $MinPts$, a new cluster with point p as a core is created. The algorithm then assembles directly density reachable points from these core points to merge all the density reachable clusters.

This process terminates when there is no new point to be added to any group. The points which do not fall in any cluster are considered to be noise or outliers.

Please click the following link for a demo of the DBSCAN algorithm:

*https://miro.medium.com/proxy/1*tc8UF-h0nQqUfLC8-0uInQ.gif*

Algorithm of the DBSCAN:

Step-1 Set values for $MinPts$ and ε, and randomly pick a data point from the dataset as the starting point.

Step-2 Find recursively all its density connected points and assign them to the same cluster as the core point.

Step-3 Randomly choose another point among the points that have not been visited in the previous steps and go to Step-2

Step-4 This process is finished when all points are visited. Those points that do not belong to any cluster are considered as noise.

Advantages of DBSCAN

- DBSCAN algorithm is robust to outliers (noise points).
- DBSCAN is great at separating high density clusters from low density clusters.
- Unlike K-means, DBSCAN does not require number of clusters to be specified *a priori*.
- DBSCAN supports clusters with irregular structures.

Disadvantages of DBSCAN

- DBSCAN does not work well for clusters of varying density.
- DBSCAN algorithm is not deterministic in the sense that it forms different clusters on different trials.
- Sometimes, choosing the value of ε can be difficult especially when the data is in higher dimensions.