K-MEANS CLUSTERING

INTRODUCTION-
What is clustering?

- **Clustering** is the classification of objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often according to some defined distance measure.

Types of clustering:

1. **Hierarchical algorithms**: these find successive clusters using previously established clusters.
   1. Agglomerative ("bottom-up"): Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters.
   2. Divisive ("top-down"): Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.
2. **Partitional clustering:** Partitional algorithms determine all clusters at once. They include:
   - ***K*-means and derivatives**
   - Fuzzy *c*-means clustering
   - QT clustering algorithm

Common Distance measures:

- *Distance measure* will determine how the *similarity* of two elements is calculated and it will influence the shape of the clusters.

  They include:

1. The Euclidean distance (also called 2-norm distance) is given by:

$$d(x,y)=\sum_{i=1}^{p}|x_i-y_i|$$

2. The Manhattan distance (also called taxicab norm or 1-norm) is given by:

$$d(x,y)=\sqrt[2]{\sum_{i=1}^{p}|x_i-y_i|^2}$$

3.The maximum norm is given by:

$$d(x,y)=\max_{1\le i\le p}|x_i-y_i|$$

4. The Mahalanobis distance corrects data for different scales and correlations in the variables.
5. Inner product space: The angle between two vectors can be used as a distance measure when clustering high dimensional data
6. Hamming distance (sometimes edit distance) measures the minimum number of substitutions required to change one member into another.

K-MEANS CLUSTERING

- The **k-means algorithm** is an algorithm to cluster *n* objects based on attributes into *k* partitions, where $k < n$.
- It is similar to the expectation-maximization algorithm for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data.
- It assumes that the object attributes form a vector space.

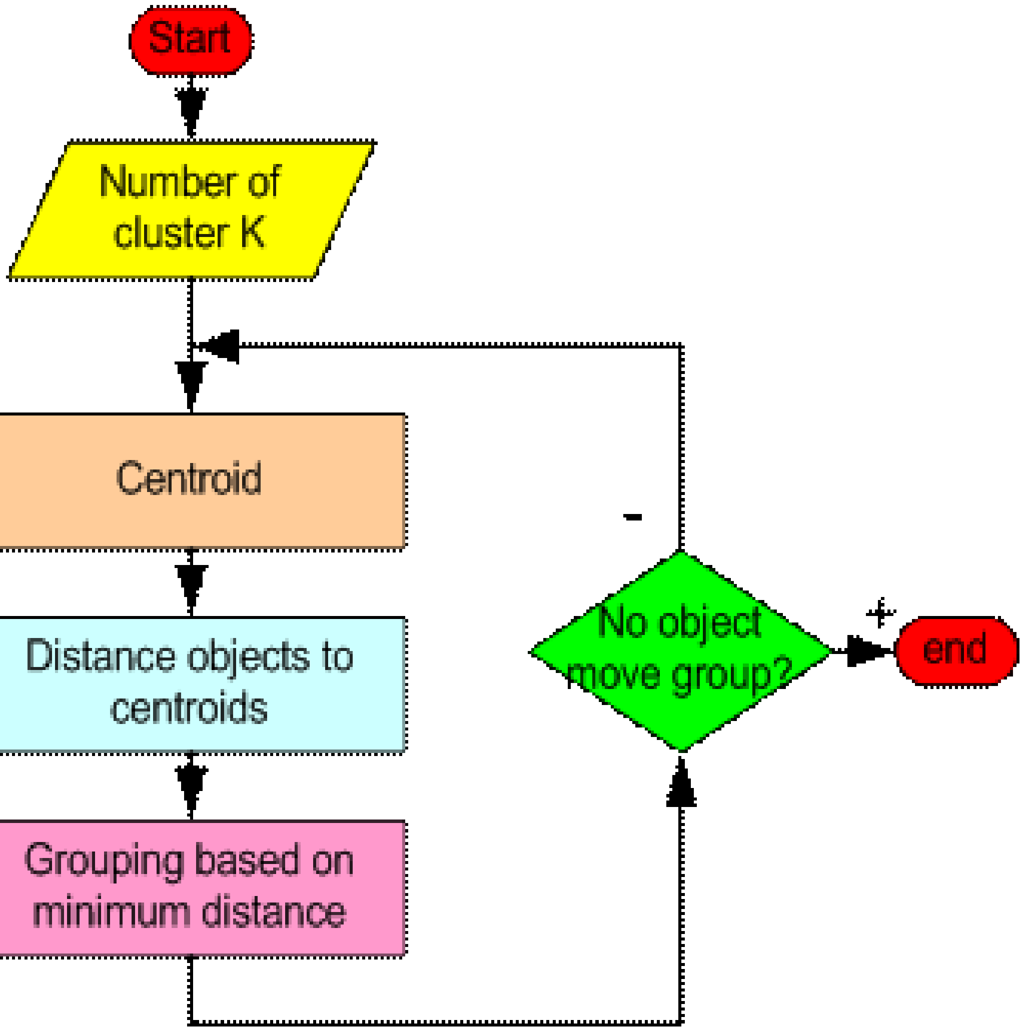
- An algorithm for partitioning (or clustering) N data points into K disjoint subsets S_j containing data points so as to minimize the sum-of-squares criterion

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} \left| x_n - \mu_j \right|^2,$$

  where x_n is a vector representing the the nth data point and u_j is the geometric centroid of the data points in S_j.

- Simply speaking k-means clustering is an algorithm to classify or to group the objects based on attributes/features into K number of group.
- K is positive integer number.
- The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid.

How the K-Mean Clustering algorithm works?


Start
Number of cluster K
Centroid
Distance objects to centroids
Grouping based on minimum distance
No object move group?
-
+
end

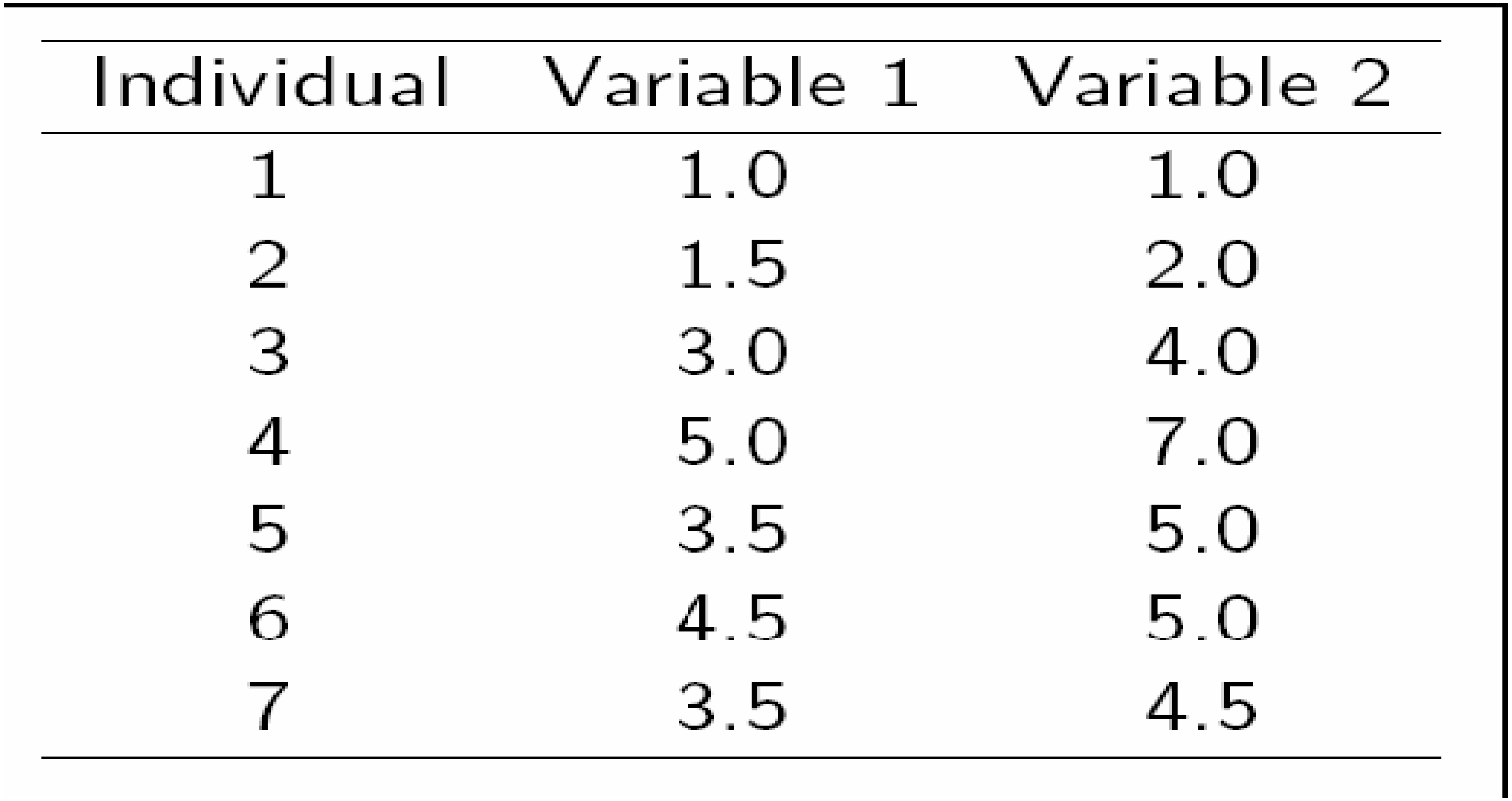
- **Step 1:** Begin with a decision on the value of k = number of clusters .
- **Step 2**: Put any initial partition that classifies the data into k clusters. You may assign the training samples randomly,or systematically as the following:

  1.Take the first k training sample as single-element clusters

  2. Assign each of the remaining (N-k) training sample to the cluster with the nearest centroid. After each assignment, recompute the centroid of the gaining cluster.

- **Step 3:** Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.
- **Step 4 .** Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

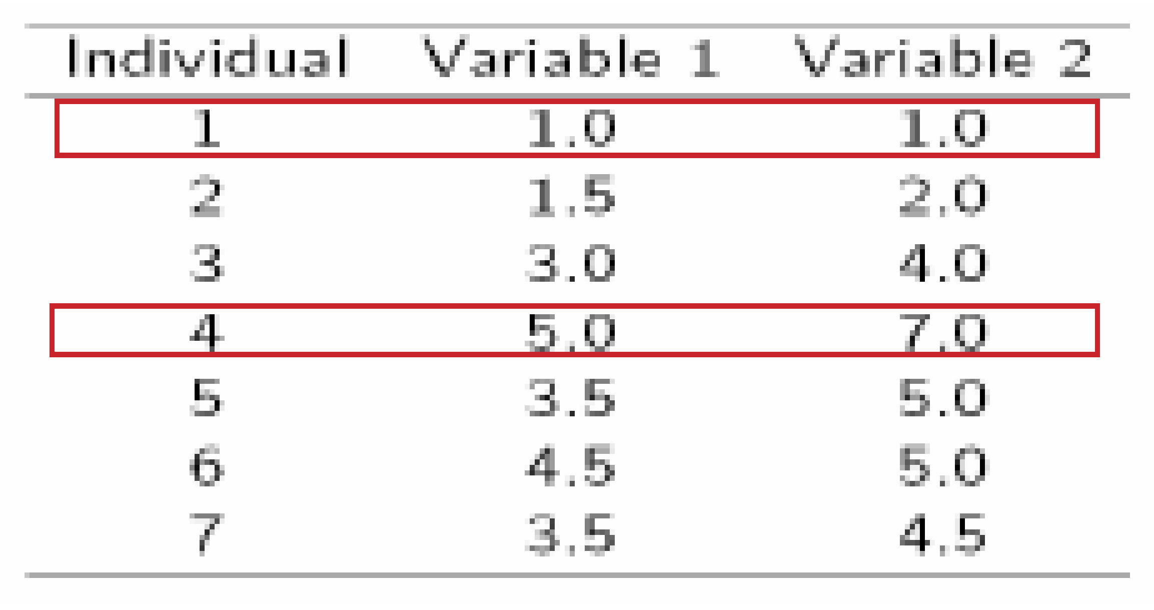
A Simple example showing the implementation of k-means algorithm (using K=2)



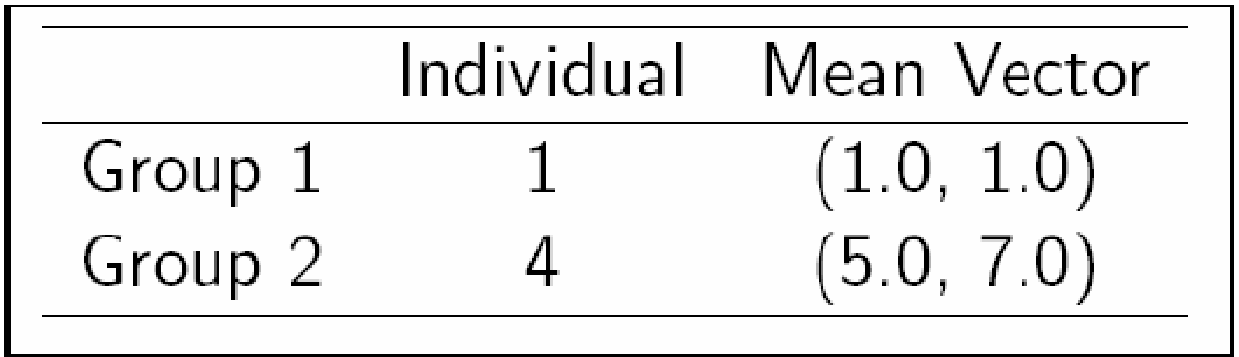
Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

Step 1:

Initialization: Randomly we choose following two centroids (k=2) for two clusters.
In this case the 2 centroid are: m1=(1.0,1.0) and m2=(5.0,7.0).



Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5



	Individual	Mean Vector
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

Step 2:

- Thus, we obtain two clusters containing:

  {1,2,3} and {4,5,6,7}.
- Their new centroids are:

$$m_1 = (\frac{1}{3}(1.0+1.5+3.0), \frac{1}{3}(1.0+2.0+4.0)) = (1.83, 2.33)$$

$$m_2 = (\frac{1}{4}(5.0+3.5+4.5+3.5), \frac{1}{4}(7.0+5.0+5.0+4.5))$$

$$= (4.12, 5.38)$$

Individual	Centroid 1	Centroid 2
1	0	7.21
2 (1.5, 2.0)	1.12	6.10
3	3.61	3.61
4	7.21	0
5	4.72	2.5
6	5.31	2.06
7	4.30	2.92

$$d(m_1, 2) = \sqrt{|1.0-1.5|^2 + |1.0-2.0|^2} = 1.12$$

$$d(m_2, 2) = \sqrt{|5.0-1.5|^2 + |7.0-2.0|^2} = 6.10$$

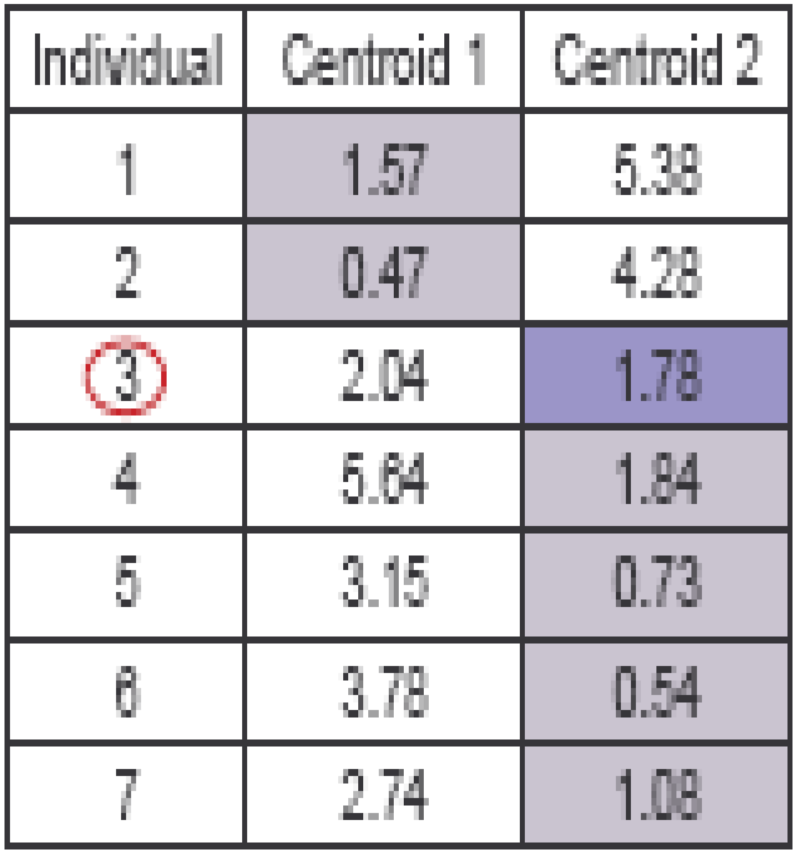
Step 3:

- Now using these centroids we compute the Euclidean distance of each object, as shown in table.

- Therefore, the new clusters are:

  {1,2} and {**3**,4,5,6,7}

- Next centroids are: m1=(1.25,1.5) and m2 = (3.9,5.1)



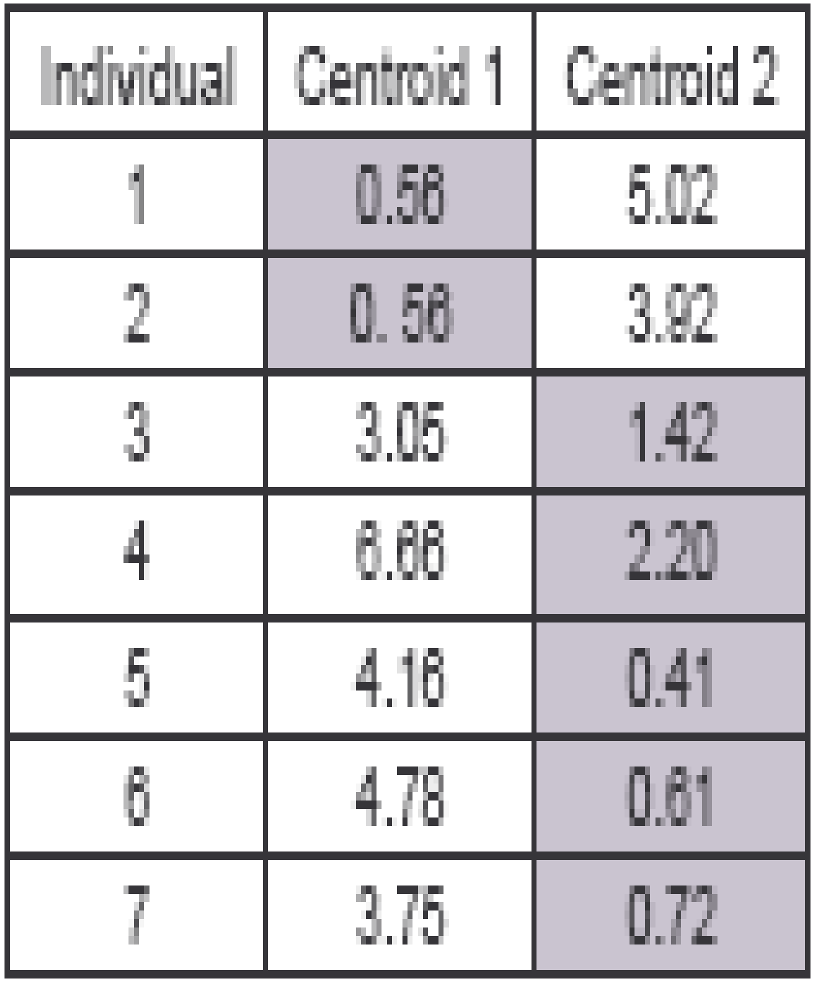
Individual	Centroid 1	Centroid 2
1	1.57	5.38
2	0.47	4.28
3	2.04	1.78
4	5.64	1.84
5	3.15	0.73
6	3.78	0.54
7	2.74	1.08

- <u>Step 4 </u>:

  The clusters obtained are:

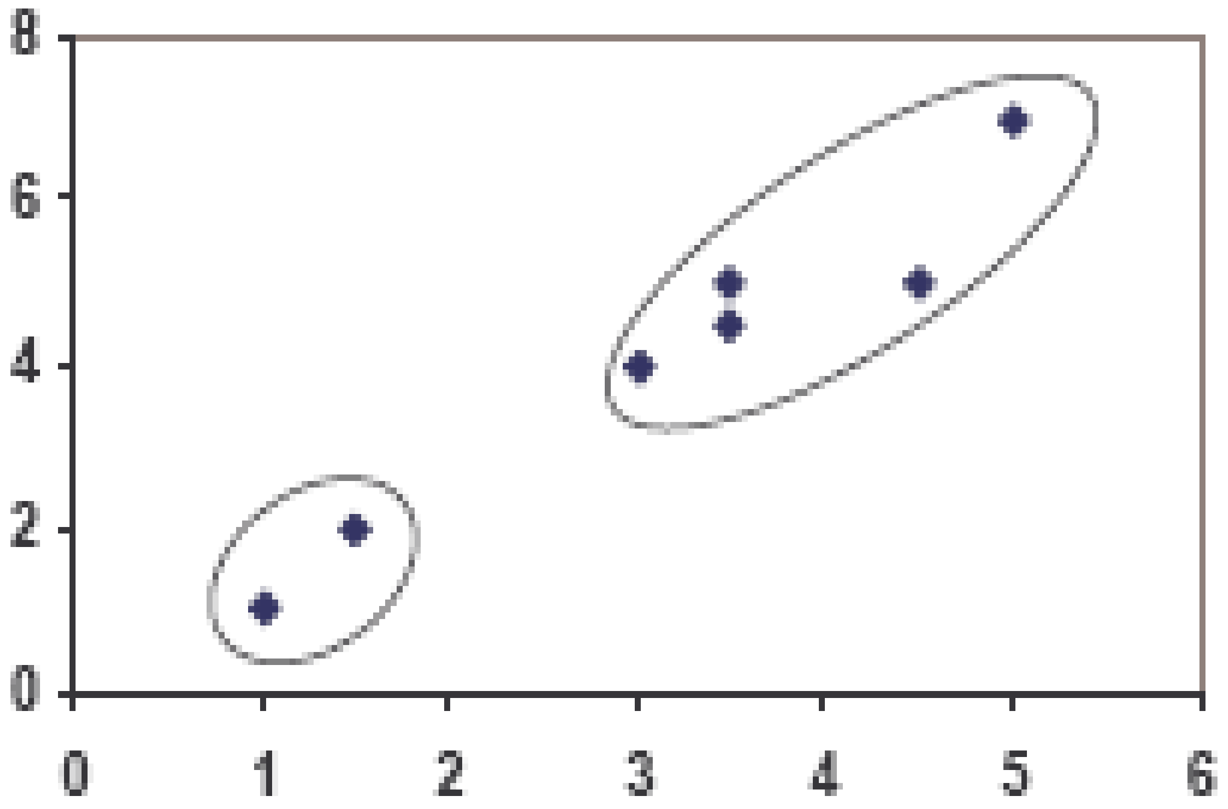
  {1,2} and {3,4,5,6,7}

- Therefore, there is no change in the cluster.
- Thus, the algorithm comes to a halt here and final result consist of 2 clusters {1,2} and {3,4,5,6,7}.

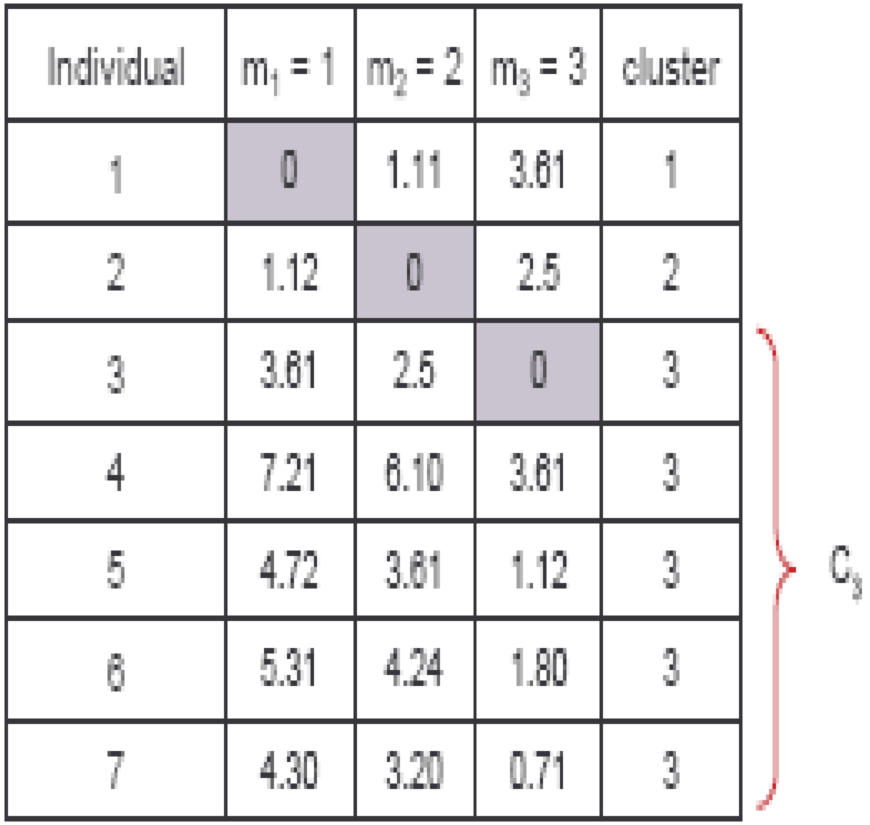


Individual	Centroid 1	Centroid 2
1	0.56	5.02
2	0. 56	3.02
3	3.05	1.42
4	6.66	2.20
5	4.16	0.41
6	4.78	0.61
7	3.75	0.72

PLOT



(with K=3)

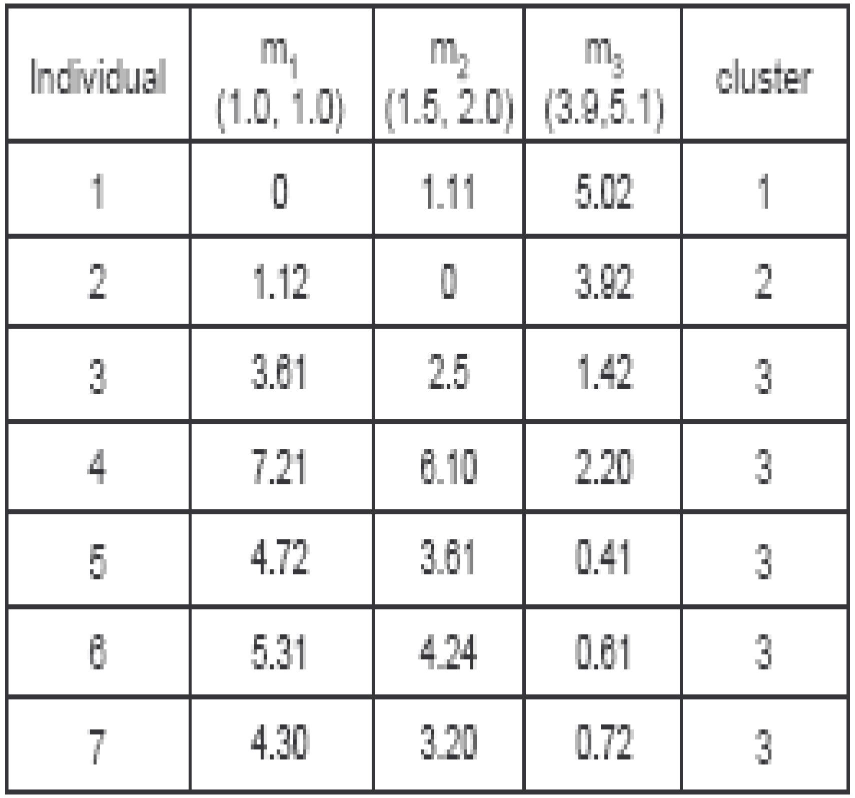


Individual	$m_1 = 1$	$m_2 = 2$	$m_3 = 3$	cluster
1	0	1.11	3.61	1
2	1.12	0	2.5	2
3	3.61	2.5	0	3
4	7.21	6.10	3.61	3
5	4.72	3.61	1.12	3
6	5.31	4.24	1.80	3
7	4.30	3.20	0.71	3

C_3

clustering with initial centroids (1, 2, 3)

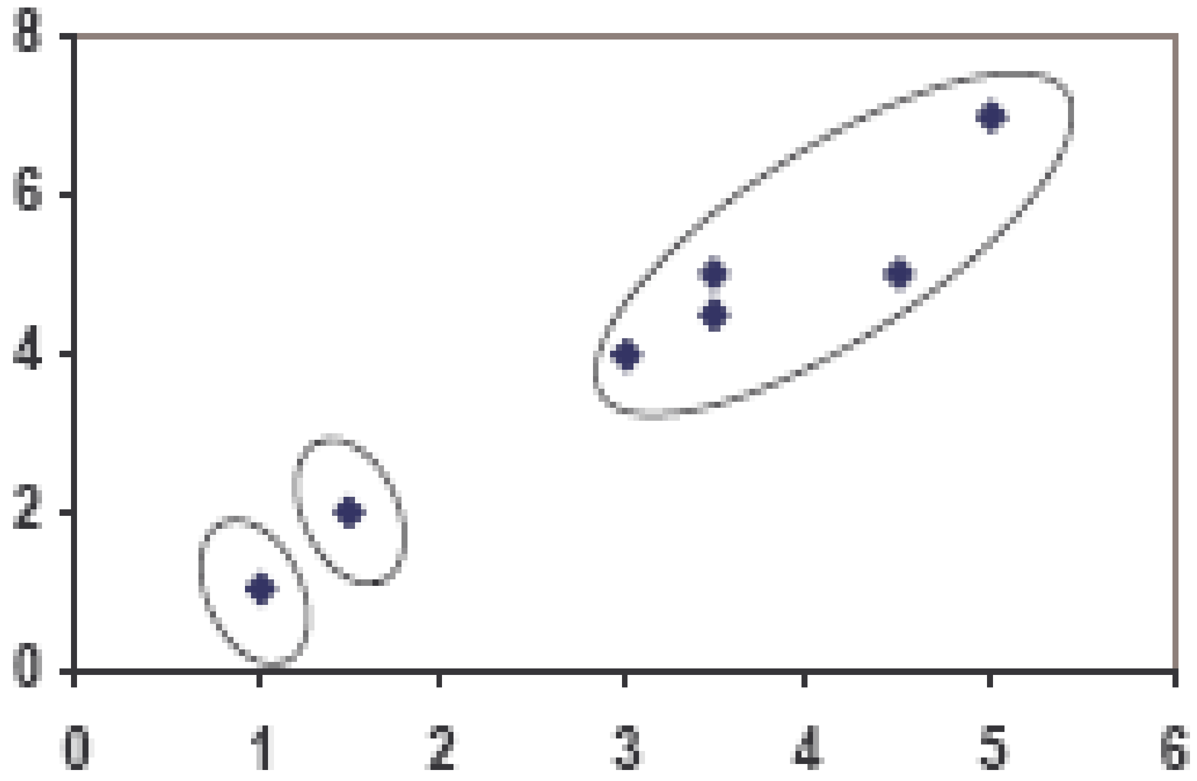
Step 1



Individual	m_1 (1.0, 1.0)	m_2 (1.5, 2.0)	m_3 (3.9,5.1)	cluster
1	0	1.11	5.02	1
2	1.12	0	3.92	2
3	3.61	2.5	1.42	3
4	7.21	6.10	2.20	3
5	4.72	3.61	0.41	3
6	5.31	4.24	0.61	3
7	4.30	3.20	0.72	3

Step 2

PLOT



Weaknesses of K-Mean Clustering

1. When the numbers of data are not so many, initial grouping will determine the cluster significantly.
2. The number of cluster, K, must be determined before hand. Its disadvantage is that it does not yield the same result with each run, since the resulting clusters depend on the initial random assignments.
3. We never know the real cluster, using the same data, because if it is inputted in a different order it may produce different cluster if the number of data is few.
4. It is sensitive to initial condition. Different initial condition may produce different result of cluster. The algorithm may be trapped in the *local optimum*.

k-medoid clustering

- **0verview of k-medoid clustering**
- K-medoids is a clustering algorithm that groups similar data points together into clusters. It is a variation of the K-means algorithm and is used to partition a set of data points into K clusters, where K is a user-defined parameter.
- The K-medoids algorithm works by selecting K data points as the initial medoids, and then assigning each data point to the closest medoid. The medoid is defined as the data point that is closest to the other data points in the cluster. The algorithm then iteratively updates the medoids to minimize the sum of distances between data points and their cluster medoids.

- One of the key differences between K-medoids and K-means is that K-medoids is more robust to outliers, as the medoid is less influenced by individual data points than the cluster centroid used in K-means. This makes K-medoids a good choice for clustering data that is categorical or binary, or for data with a non-Euclidean distance metric.
- The K-medoids algorithm is computationally slower than K-means, as it requires finding the closest data point to the medoid. However, it is more flexible in terms of the distance metric used, as it can handle non-Euclidean distance metrics.

K-medoids is a useful clustering algorithm for grouping similar data points into clusters and is particularly useful for categorical or binary data, or data with a non-Euclidean distance metric.

K medoids algorithm

- The K-medoids algorithm is a clustering algorithm that groups similar data points into K clusters, where K is a user-defined parameter. The algorithm works as follows:
- Step 1->**Initialization**: Select K data points randomly as the initial medoids.
- Step 2->**Assignment** : For each data point, calculate the distance to all K medoids and assign the data point to the closest medoid. This creates K clusters, each with a medoid and a set of data points assigned to it.
- Step 3->**Update:**For each cluster, calculate the cost of replacing its medoid with each of its data points. If replacing the medoid with a data point results in a lower cost, update the medoid to that data point. Repeat this process until no further improvements can be made.
- Step 4->**Repeat steps 2 and 3** until the medoids no longer change or a maximum number of iterations is reached.

- Step 5->**Output** The final set of K medoids and the data points assigned to each cluster.

The K-medoids algorithm aims to minimize the sum of distances between data points and their cluster medoids. The distance metric used can be Euclidean, Manhattan, or any other user-defined distance metric. The choice of distance metric depends on the characteristics of the data and the requirements of the application.

Advantages

- **Robust to outliers:** K-medoids is less sensitive to outliers than K-means, as the medoid is less influenced by individual data points than the cluster centroid used in K-means.
- **Flexibility in distance metrics:** K-medoids can handle non-Euclidean distance metrics, making it a good choice for clustering data with different types of features or data with a non-linear structure.
- **Simple implementation:** K-medoids is a simple and easy-to-understand algorithm, making it a good choice for clustering problems with a small number of data points or for small-scale applications.
- **Faster convergence:** K-medoids can converge faster than K-means, as it only requires calculating the distances between the data points and the medoids.

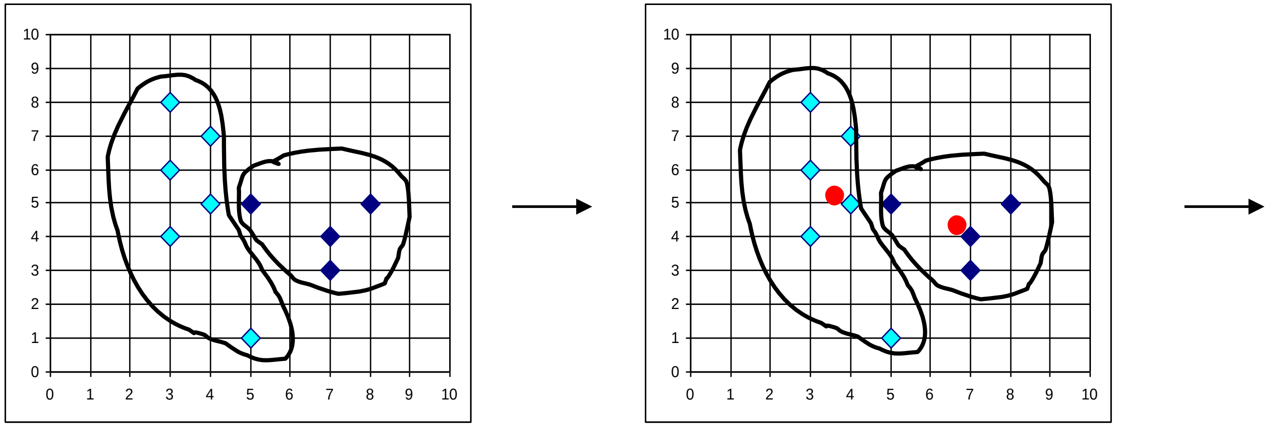
Disadvantages

- **Computational cost:** K-medoids is computationally more expensive than K-means, as it requires finding the closest data point to the medoid for each data point.
- **Local minima:** K-medoids is prone to getting stuck in local minima, meaning that it may not find the optimal solution for the clustering problem.
- **Difficult to scale:** K-medoids is not well suited for large-scale clustering problems, as the computational cost increases with the number of data points.
- **Sensitive to initial medoids:** The quality of the solution obtained by K-medoids is sensitive to the initial medoids selected, so care must be taken in selecting the initial medoids.

Text Clustering

Bisecting k-means [Steinbach, Karypis & Kumar 2000]

K-means



Bisecting k-means

- Partition the database into 2 clusters
- **Repeat**: partition the largest cluster into 2 clusters . . .
- **Until** *k* clusters have been discovered

Text Clustering

Bisecting k-means

Two types of clusterings

- Hierarchical clustering
- Flat clustering: any cut of this hierarchy

Distance Function

- Cosine measure (similarity measure)

$$s(\alpha, \beta) = \frac{\langle g(c(\alpha)), g(c(\beta)) \rangle}{\| g(c(\alpha)) \| \cdot \| g(c(\beta)) \|}$$