

CHAPTER 7

Clustering

Learning Objectives

At the end of this chapter, you will be able to:

- Define clustering
- Describe the commonly used clustering algorithms

7.1 INTRODUCTION TO CLUSTERING

Clustering refers to the process of arranging or organizing objects according to specific criteria. It plays a crucial role in uncovering concealed knowledge in large data sets. Clustering involves dividing or grouping the data into smaller data sets based on similarities/dissimilarities. Depending on the requirements, this grouping can lead to various outcomes, such as partitioning of data, data re-organization, compression of data and data summarization.

7.1.1 Partitioning of Data

Clustering of data is a crucial aspect of efficient data access in database-based applications.

EXAMPLE 1: To illustrate, let us take the example of an EMPLOYEE data table that contains 30,000 records of fixed length. We assume that there are 1000 distinct values of DEPT_CODE and that the employee records are evenly distributed among these values.

If this data table is clustered by DEPT_CODE, accessing all the employees of the department with DEPT_CODE = '15' requires $\log_2(1000) + 30$ accesses. The first term involves accessing the index table that is constructed using DEPT_CODE, which is achieved through binary search. The second term involves fetching 30 (that is, $30000/1000$) records from the clustered (that is, grouped based on DEPT_CODE) employee table, which is indicated by the fetched entry in the index table.

Without clustering, accessing 30 employee records from a department would require, on average, $30 \times 30000/2$ accesses.

7.1.2 Data Re-organization

EXAMPLE 2: We can demonstrate the concept of data re-organization using the binary pattern matrix displayed in Table 7.1.

TABLE 7.1 Row data

Pattern	f1	f2	f3	f4	f5	f6
1	1	0	1	0	1	0
2	0	1	0	1	0	1
3	1	0	1	0	1	0
4	0	1	0	1	0	1

This data set comprises four binary patterns, each with six dimensions or features labelled f1 to f6 and assigned a unique ID ranging from 1 to 4. By clustering the rows, we can obtain the re-organized data set shown in Table 7.2. This process groups similar patterns together based on their row-wise similarities.

TABLE 7.2 Data re-organization using row as the criterion

Pattern	f1	f2	f3	f4	f5	f6
1	1	0	1	0	1	0
3	1	0	1	0	1	0
2	0	1	0	1	0	1
4	0	1	0	1	0	1

Further, by clustering the re-organized data set from Table 7.2 by column and grouping similar columns together, we obtain the data set displayed in Table 7.3.

TABLE 7.3 Data re-organization using rows and columns

Pattern	f1	f2	f3	f4	f5	f6
1	1	1	1	0	0	0
3	1	1	1	0	0	0
2	0	0	0	1	1	1
4	0	0	0	1	1	1

This table reveals the hidden structure in the data. The cluster comprising patterns 1 and 3 can be described using the conjunction $f1 \wedge f3 \wedge f5$. Likewise, the second cluster comprising patterns 2 and 4 can be described using $f2 \wedge f4 \wedge f6$.

7.1.3 Data Compression

Clustering can also assist in data compression by reducing the time complexity of accessing the data features and minimizing the space required to store them.

EXAMPLE 3: To demonstrate, consider the data set shown in Table 7.1. Through clustering, we can compress the data set, resulting in the table shown in Table 7.4. Note that the frequent pattern (FP) tree and pattern count (PC) tree structures are some examples for achieving such compression of data.

TABLE 7.4 Compressed data

Pattern	f1	f2	f3	f4	f5	f6	Count
1, 3	1	0	1	0	1	0	2
2, 4	0	1	0	1	0	1	2

7.1.4 Summarization

The objective of data summarization is to extract a representative subset of samples from a large data set. The purpose is to simplify and expedite analysis on a smaller data set.

EXAMPLE 4: For instance, when dealing with a data set of the marks of 100 students in a subject, statistical measures like mean (the numerical average of the marks), mode (the most frequently repeated marks) and median (the value in the middle of all the marks when the marks are ranked in order) can provide summarized information. Such summarized information can be derived through clustering.

7.1.5 Matrix Factorization

It is possible to view clustering as matrix factorization.

Let there be n data points in an l -dimensional space. We can represent it as a matrix $X_{n \times l}$. It is possible to approximate X as a product of two matrices $B_{n \times K}$ and $C_{K \times l}$. So, $X \approx BC$, where B is the cluster assignment matrix and C is the representatives matrix.

EXAMPLE 5: Consider the data matrix

$$X = \begin{bmatrix} 6 & 6 & 6 \\ 6 & 6 & 8 \\ 2 & 4 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$

There are four patterns in a three-dimensional space. If we use the leader algorithm with a threshold of 3 units, we get two clusters: (6,6,6) and (6,6,8) that belong to cluster 1 with (6,6,6) as its leader and (2,4,2) and (2,2,2) that belong to cluster 2 with (2,4,2) as its leader.

So, we have $B = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$. It indicates that the first pattern, (6,6,6), is assigned to cluster 1.

Correspondingly in B we have a 1 in the first row and first column. Matrix B has four rows

corresponding to four patterns in X ; B has two columns corresponding to two clusters. Matrix C will have K rows if there are K clusters and the i^{th} row is the representative of the i^{th} cluster.

In the current example, $C = \begin{bmatrix} 6 & 6 & 6 \\ 2 & 4 & 2 \end{bmatrix}$. So the resulting matrix factorization is given by

$$\begin{bmatrix} 6 & 6 & 6 \\ 6 & 6 & 8 \\ 2 & 4 & 2 \\ 2 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} 6 & 6 & 6 \\ 2 & 4 & 2 \end{bmatrix}$$

Leader is a hard clustering algorithm where each pattern is assigned fully, that is, with a value 1, to a single cluster. So, each row of the B matrix will have one 1 and the rest as 0s.

It is possible to have soft clustering. In this case, we can have multiple non-zero entries in the range $[0,1]$ in each row, indicating that a pattern is assigned to more than one cluster. The sum of the entries in a row is 1.

Instead of the leader, if we use the centroid of points in the cluster as its representative, then $C = \begin{bmatrix} 6 & 6 & 7 \\ 2 & 3 & 2 \end{bmatrix}$. There is no change in B . Note that the centroid of $\{(6,6,6), (6,6,8)\}$ is $(6,6,7)$ and the centroid of the second cluster $\{(2,4,2), (2,2,2)\}$ is $(2,3,2)$.

7.2 CLUSTERING OF PATTERNS

Clustering is a technique that involves grouping a set of patterns, resulting in the creation of cohesive clusters or groups from a given collection of patterns. The process of clustering aims to group similar patterns together while keeping dissimilar patterns separate. Figure 7.1 illustrates the clustering of a two-dimensional data set (represented by $f1$ and $f2$), where three clusters are visually identifiable.

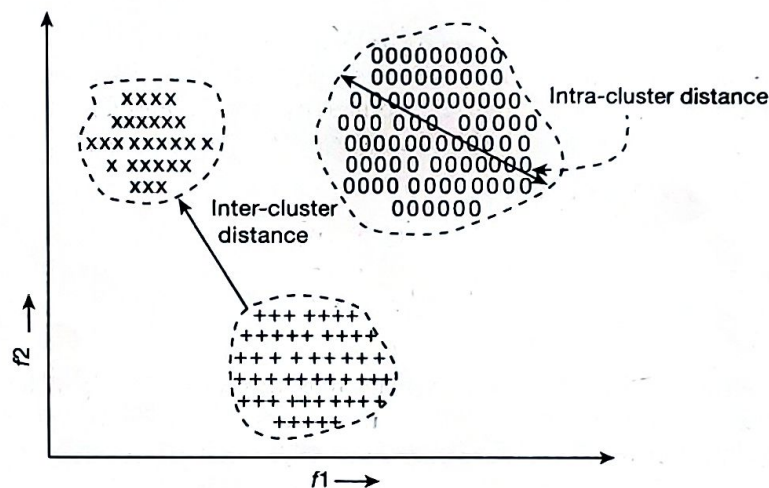


FIG. 7.1 Inter- and intra-cluster distances

The clustering process may ensure that the distance between any two points within the same cluster (intra-cluster distance), as measured by a dissimilarity measure such as Euclidean distance, is smaller than the distance between any two points belonging to different clusters (inter-cluster distance).

This indicates that the similarity between points within a cluster is higher than the similarity between clusters. We illustrate this idea with the following example.

EXAMPLE 6: Let us consider a two-dimensional data set with 11 data points having two features f_1 and f_2 as listed in Table 7.5.

TABLE 7.5 Data set with two features

Data point	f_1	f_2
x_1	1	1
x_2	1	3
x_3	2	2
x_4	3	1
x_5	3	3
x_6	4	7
x_7	5	7
x_8	6	9
x_9	7	1
x_{10}	7	3
x_{11}	9	1

Any two points are placed in the same cluster if the distance between them is lower than a certain threshold. In this example, the squared Euclidean distance is used to measure the distance between the points, and a threshold of 10 units is set to cluster them. The squared Euclidean distance (d) between two points, x_i and x_j , is calculated as follows:

$$d(x_i, x_j) = \sum_{k=1}^l (x_i(k) - x_j(k))^2,$$

where l represents the dimensionality of the points.

In this example, since we are dealing with two-dimensional data points, l equals 2. Using this formula, the squared Euclidean distance between all pairs of points can be found and is presented in Table 7.6.

In Table 7.6, the clusters are clearly visible within the matrix itself. The table contains three sub-matrices of sizes 5×5 , 3×3 and 3×3 that meet the condition of having a maximum value of 10 in any entry. For instance, the sub-matrix of size 5×5 , corresponding to the first five patterns (say, Cluster_A = $\{x_1, x_2, x_3, x_4, x_5\}$) has values ranging from 0 to 8, with every other entry in the first five rows exceeding 10. Similarly, it can be deduced that patterns x_6, x_7 and x_8 belong to the second cluster (Cluster_B = $\{x_6, x_7, x_8\}$) while patterns x_9, x_{10} and x_{11} belong to the third cluster (Cluster_C = $\{x_9, x_{10}, x_{11}\}$).

One can observe from Table 7.6 that none of the data points is present in more than one cluster. Such a clustering is called *hard clustering*, otherwise it is known as *soft or overlapping clustering*. We discuss soft clustering in Section 7.6.

TABLE 7.6 Squared Euclidean distance between pairs of data points listed in Table 7.4

Data point	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}
x_1	0	4	2	4	8	45	52	89	36	40	64
x_2	4	0	2	8	4	25	32	61	40	36	68
x_3	2	2	0	2	2	29	34	65	26	26	50
x_4	4	8	2	0	4	37	40	73	16	20	36
x_5	8	4	2	4	0	17	20	45	20	16	40
x_6	45	25	29	37	17	0	1	8	45	25	61
x_7	52	32	34	40	20	1	0	5	40	20	52
x_8	89	61	65	73	45	8	5	0	65	37	73
x_9	36	40	26	16	20	45	40	65	0	4	4
x_{10}	40	36	26	20	16	25	20	37	4	0	8
x_{11}	64	68	50	36	40	61	52	73	4	8	0

Let $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ be n data points which are mapped to C clusters, $\mathcal{C} = \{X_1, X_2, \dots, X_C\}$ such that

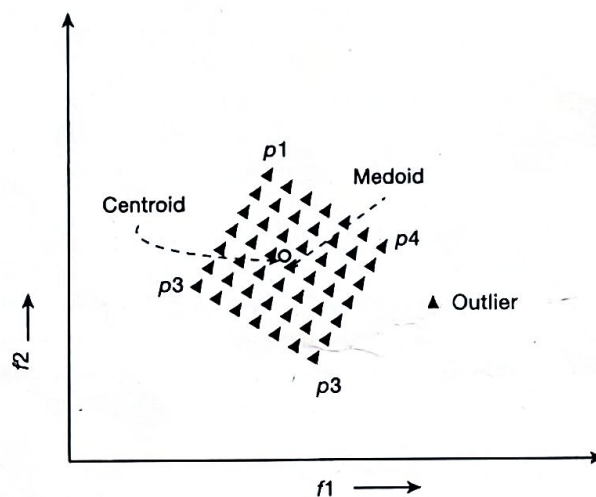
- $\bigcup_{i=1}^C X_i = \mathcal{X}$
- $X_i \neq \phi, \forall i \in \{1, 2, \dots, C\}$

If $\forall i, j, i \neq j, x_i \cap x_j = \phi$, then the clustering is **hard**, else it is **soft**.

7.2.1 Data Abstraction

Clustering is a useful method for data abstraction, and it can be applied to generate clusters of data points that can be represented by their centroid, medoid, leader or some other suitable entity.

The centroid is computed as the sample mean of the data points in a cluster, and it is given by $\frac{1}{n_C} \times \sum (x_i \in C)$, where n_C is the number of patterns in the cluster C . Note that it may not coincide with any specific data point, as shown in Fig. 7.2.

**FIG. 7.2** Visualization of centroid and medoid

The medoid is the point that minimizes the sum of distances to all other points in the cluster. Figure 7.2 illustrates this process, where an outlier that is located far away from the data points in the cluster is also present. The centroid can shift dramatically based on the position of the outlier, while the medoid remains stable within the boundaries of the original cluster. Therefore, clustering algorithms that utilize medoids are more resilient to noisy patterns or outliers.

EXAMPLE 7: Consider the data set shown in Table 7.5 and the cluster thus generated given in Table 7.6. The centroid of the clusters, Cluster_A (that is, μ_A), Cluster_B (that is, μ_B) and Cluster_C (that is, μ_C), are given below:

$$\mu_A = \frac{1}{5} [(1, 1) + (1, 3) + (2, 2) + (3, 1) + (3, 3)] = \left(\frac{1+1+2+3+3}{5}, \frac{1+3+2+1+3}{5} \right) = (2, 2)$$

$$\mu_B = \frac{1}{3} [(4, 7) + (5, 7) + (6, 9)] = \left(\frac{4+5+6}{3}, \frac{7+7+9}{3} \right) = (5, 7.67)$$

$$\mu_C = \frac{1}{3} [(7, 1) + (7, 3) + (9, 1)] = \left(\frac{7+7+9}{3}, \frac{1+3+1}{3} \right) = (7.67, 1.67)$$

For the same clusters, the medoids are given below:

$$m_A = (2, 2), m_B = (5, 7) \text{ and } m_C = (7, 3)$$

We know that the medoid of a group of points is defined as a data point within the cluster that is located at the centre, where the total distance from the points in the cluster is at its minimum. Using the squared Euclidean distance as the metric for calculating distance, the distance of the points from m_A , m_B and m_C can be determined as follows:

For Cluster_A: Considering X_5 as the medoid (m_A):

$$d(x_1, m_A) = 2$$

$$d(x_2, m_A) = 2$$

$$d(x_3, m_A) = 0$$

$$d(x_4, m_A) = 2$$

$$d(x_5, m_A) = 2$$

The sum of the distances = 8. Note that if you choose any other data point, say x_1 as m_A , then the squared Euclidean distance is:

$$d(x_1, m_A) = 0$$

$$d(x_2, m_A) = 4$$

$$d(x_3, m_A) = 2$$

$$d(x_4, m_A) = 4$$

$$d(x_5, m_A) = 8$$

The sum of the distances = 18. Similarly, one can find that on choosing any point other than x_3 as the medoid, the corresponding sum of the distances is greater than 8.

For Cluster_B: Considering x_7 as medoid (m_B):

$$d(x_6, m_B) = 1$$

$$d(x_7, m_B) = 0$$

$$d(x_8, m_B) = 5$$

The sum of the distances = 6. Note that if you choose any other data point as m_B , then the squared Euclidean distance is greater than 6. So x_7 is the medoid of Cluster_B.

For Cluster_C: Considering x_9 as medoid (m_C):

$$d(x_9, m_C) = 0$$

$$d(x_{10}, m_C) = 4$$

$$d(x_{11}, m_C) = 4$$

The sum of the distances = 8. Note that if you choose any other data point as m_C , then the squared Euclidean distance is greater than 8. So x_9 is the medoid of Cluster_C.

In the above example, even though centroid and medoid are the same for Cluster_A, they may not be the same as shown for Cluster_B and Cluster_C.

The following example shows that the medoid of a cluster is more robust to the addition of outliers.

EXAMPLE 8: Consider the cluster of three data points given below:

$$x_1 = (4, 1) \quad x_2 = (5, 1) \quad x_3 = (6, 3)$$

The centroid μ for this cluster is

$$\mu = \left(\frac{4+5+6}{3}, \frac{1+1+3}{3} \right) = (5, 1.67)$$

and the medoid m for this cluster can be obtained from Table 7.7.

TABLE 7.7 Squared Euclidean distance between the data points

	x_1	x_2	x_3	Σ
x_1	0	1	8	9
x_2	1	0	5	6
x_3	8	5	0	13

In Table 7.7, the data point x_2 has the minimum squared Euclidean distance with all other data points (refer the last column of the table). So, x_2 (that is, (5,1)) is the medoid.

If we add an outlier point, $x_4 = (20, 1)$, then the new centroid is

$$\hat{\mu} = \left(\frac{4+5+6+20}{4}, \frac{1+1+3+1}{4} \right) = (8.75, 1.5)$$

and the new medoid \hat{m} for this cluster can be obtained from Table 7.8. In Table 7.8, the data point x_3 has the minimum squared Euclidean distance with all other data points (refer the last column of the table). So, x_3 (that is, (6,1)) is the medoid.

One can observe that because of x_4 (an outlier in this case), the centroid is shifted from $\mu = (5, 1.67)$ to $\hat{\mu} = (8.75, 1.5)$ and the medoid is shifted from $m = (5, 1)$ to $\hat{m} = (6, 3)$. But the change from m to \hat{m} is very small when compared to the change from μ to $\hat{\mu}$. So, the medoid of the cluster is more robust to the addition of outliers.

TABLE 7.8 Squared Euclidean distance between the data points

	x_1	x_2	x_3	x_4	Σ
x_1	0	1	8	256	265
x_2	1	0	5	225	231
x_3	8	5	0	200	213
x_4	256	225	200	0	681

Note that in the above examples, we have used centroid and medoid as the representatives or descriptions of the cluster. However, it is feasible to have more than one or multiple representative elements per cluster.

As an instance, a cluster can be represented by four extreme points, namely, p_1 , p_2 , p_3 and p_4 , as illustrated in Fig. 7.2. Additionally, it is possible to use a logical description to depict a cluster. For instance, a cluster can be characterised by $(\text{dept_name} = \text{"Computer Sc"} \vee \text{dept_name} = \text{"Mech Engg"}) \wedge \text{Salary} > 200,000$.

Clusters are useful in several decision-making situations like classification, prediction, etc. However, when the number of cluster representatives obtained is smaller than the number of input patterns, there is a reduction in computation while performing classification or prediction activity. The following example demonstrates this.

EXAMPLE 9: Consider the data set shown in Table 7.9. Let $(2, 4)$ be a test pattern which needs to be classified using the nearest neighbor classifier on the 11 labelled patterns of Table 7.9.

TABLE 7.9 Data set with two features and corresponding labels

Data point	f1	f2	Class label
x_1	1	1	A
x_2	1	3	A
x_3	2	2	A
x_4	3	1	A
x_5	3	3	A
x_6	4	7	B
x_7	5	7	B
x_8	6	9	B
x_9	7	1	B
x_{10}	7	3	B
x_{11}	9	1	B

Note that $(2, 4)$ is the nearest neighbor of $(3, 3)$ with a squared Euclidean distance of 2 units between them. This can be obtained by computing 11 distance values – calculating the squared Euclidean distance between the test pattern $(2, 4)$ and each of the 11 labelled patterns. So, $(2, 4)$ is classified as belonging to class “A” because its nearest neighbor $(3, 3)$ belongs to “A”.

However, by clustering the 11 patterns using a suitable clustering algorithm and representing each resulting cluster by its centroid, we can reduce the number of distance values to be computed, from the number of labelled patterns to the number of cluster representatives. This may be illustrated as follows.

Let the 11 patterns be clustered using the same criterion as the one used in Example 7, that is, the squared Euclidean distance between any two patterns placed in the same cluster should be within a threshold of 10 units. This results in three clusters: one from class "A" and two from class "B". The cluster corresponding to class "A" is:

$$\text{Cluster_A} = \{(1, 1), (1, 3), (2, 2), (3, 1), (3, 3)\} \text{ with centroid } \mu_A = (2, 2)$$

The two clusters corresponding to class "B" are:

$$\text{Cluster_B} = \{(4, 7), (5, 7), (6, 9)\} \text{ with centroid } \mu_B = (5, 7.67)$$

$$\text{Cluster_C} = \{(7, 1), (7, 3), (9, 1)\} \text{ with centroid } \mu_C = (7.67, 1.67)$$

So, instead of using the 11 labelled patterns, if one uses the centroids of the three clusters as the labelled data, then there is a reduction in the data to be analysed by a factor of 3. The reduced labelled data is given in Table 7.10.

TABLE 7.10 Reduced data set with only cluster representatives

Data point	f1	f2	Class label
μ_A	2	2	A
μ_B	5	7.67	B
μ_C	7.67	1.67	B

By referring to Table 7.10, it is possible to determine that the nearest neighbor of the point (2, 4) belongs to class "A", specifically the point (2, 2). As a result, the test pattern is classified as belonging to class "A" by making only three distance computations between the test pattern and each of the three centroids. Moreover, it should be noted that for the distance computations, only the centroids of the three clusters need to be stored, leading to a reduction in both time and space.

In this example, since we need to compute only three distances instead of 11 and need to store only three centroids out of 11 patterns, there is a 73% reduction in both time and space.

It could be argued that clustering the labelled data and computing the centroids may require a significant amount of time and space. However, it is possible to perform the clustering process as soon as the labelled data becomes available, prior to the classification stage. Furthermore, computation of the cluster centroid is a one-time affair. After obtaining the representative centroids, we can use them as labelled data to classify new test patterns, resulting in a significant reduction in both time and space requirements.

This clustering-based classification approach is particularly valuable for solving large-scale pattern classification problems that commonly arise in machine learning and also in dealing with *class imbalance*, where patterns in the majority class are clustered to have a smaller number of representatives from the majority class.

7.2.2 Clustering Algorithms

A wide variety of clustering algorithms are in use today. Figure 7.3 shows the taxonomy of clustering algorithms.

Clustering algorithms can be classified into hard and soft clustering, depending on whether clusters share data points or not. Hard clustering algorithms generate a partition of the given data set, while hierarchical algorithms generate a nested sequence of partitions. On the other hand, soft clustering algorithms utilize fuzzy sets, rough sets or evolutionary algorithms.

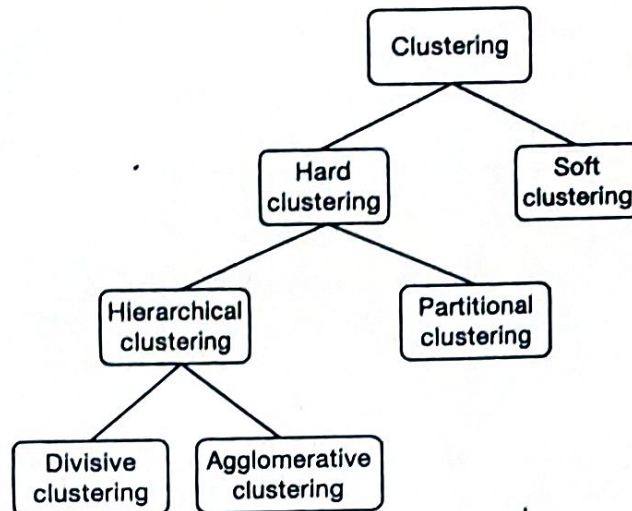


FIG. 7.3 Classification of clustering algorithms

Hierarchical Algorithms

The process of generating a sequence of data partitions using hierarchical algorithms can be represented using a tree structure called a **dendrogram**. There are two types of hierarchical algorithms – divisive and agglomerative.

Divisive algorithms follow a top-down approach, where a single cluster with all the patterns is split into smaller clusters at each step until there is only one pattern in each cluster or a collection of singleton clusters. On the other hand, agglomerative algorithms follow a bottom-up approach, where n singleton clusters are created for n input patterns. At each level, the two most similar clusters are merged to reduce the size of the partition by 1.

Once two patterns are placed in the same cluster in agglomerative algorithms, they remain in the same cluster in subsequent levels. Similarly, once two patterns are placed in different clusters in divisive algorithms, they remain in different clusters in subsequent levels.

In the next sections, we discuss divisive clustering, agglomerative clustering and partitional clustering in detail.

7.3 DIVISIVE CLUSTERING

Divisive algorithms are either polythetic, where the division is based on more than one feature, or monothetic, when only one feature is considered at a time. The **polythetic scheme** is based on finding all possible 2-partitions of the data and choosing the best among them. If there are n patterns, the number of distinct 2-partitions is given by $\frac{2^n - 2}{2} = 2^{n-1} - 1$.

EXAMPLE 10: For example, if the data set contains three patterns, x_1 , x_2 and x_3 , the possible 2-partitions among these patterns are given in Table 7.11.

TABLE 7.11 All possible 2-partition sets for the patterns x_1, x_2 and x_3

Possibilities	Non-empty subset	Its complement	2-partition
1	$\{x_1\}$	$\{x_2, x_3\}$	$\{x_1, \{x_2, x_3\}\}$
2	$\{x_2\}$	$\{x_1, x_3\}$	$\{x_2, \{x_1, x_3\}\}$
3	$\{x_3\}$	$\{x_1, x_2\}$	$\{x_3, \{x_1, x_2\}\}$
4	$\{x_2, x_3\}$	$\{x_1\}$	$\{\{x_2, x_3\}, x_1\}$
5	$\{x_1, x_3\}$	$\{x_2\}$	$\{\{x_1, x_3\}, x_2\}$
6	$\{x_1, x_2\}$	$\{x_3\}$	$\{\{x_1, x_2\}, x_3\}$

Note that in Table 7.11, the 2-partitions under the possibilities 1 and 4, 2 and 5 and 3 and 6 are repetitions. Further, the subsets ϕ and $\{x_1, x_2, x_3\}$ can be ignored. This is because ϕ is an empty set and $\{x_1, x_2, x_3\}$ is an improper subset. So, the number of possible 2-partitions is $\frac{2^3-2}{2} = 3$.

Among all possible 2-partitions, the partition with the least sum of the sample variances of the two clusters is chosen as the best. From the resulting partition, the cluster with the maximum sample variance is selected and is split into an optimal 2-partition. This process is repeated till we get singleton clusters. If a collection of patterns (data points) is split into two clusters with p patterns x_1, \dots, x_p in one cluster and q patterns y_1, \dots, y_q in the other cluster, with the centroids of the two clusters being $C1$ and $C2$, respectively, then the sum of the sample variances will be

$$\sum_{i=1}^p (x_i - C1)^2 + \sum_{j=1}^q (y_j - C2)^2$$

EXAMPLE 11: Consider the data set containing eight points (patterns) with two features as shown in Table 7.12. Figure 7.4 shows the visual representation of these eight data patterns.

TABLE 7.12 Example of data patterns with two features

Patterns	f1	f2
x_1	5.5	5.5
x_2	7	6.5
x_3	7	5.5
x_4	10	6
x_5	10.75	6
x_6	10	8
x_7	10.5	8
x_8	7	8

Figure 7.5 shows the dendrogram corresponding to the divisive clustering using the procedure discussed above. The top of the dendrogram shows a single cluster consisting of all eight data points. By evaluating all possible 2-partitions (out of $2^7 - 1 = 127$), the best 2-partition $\{\{x_1, x_2, x_3, x_8\}, \{x_4, x_5, x_6, x_7\}\}$ is obtained and shown in the dendrogram.

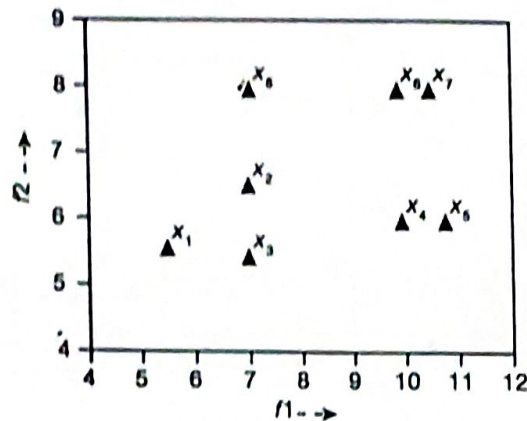


FIG. 7.4 Visual representation of data patterns listed in Table 7.12

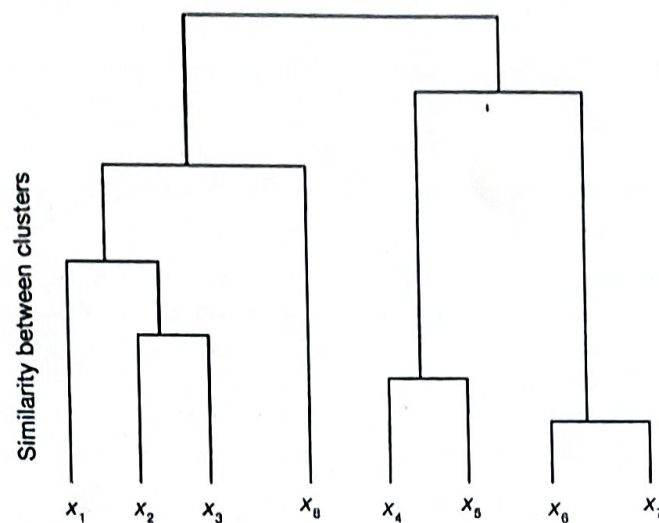


FIG. 7.5 Divisive clustering for polythetic clustering for the data points listed in Table 7.12

The cluster $\{x_4, x_5, x_6, x_7\}$ is then selected to split into two clusters, $\{x_4, x_5\}$ and $\{x_6, x_7\}$. After the split, the three clusters at that level of the dendrogram are $\{x_1, x_2, x_3, x_8\}$, $\{x_4, x_5\}$ and $\{x_6, x_7\}$.

At the subsequent levels, the cluster $\{x_1, x_2, x_3, x_8\}$ is split into $\{x_1, x_2, x_3\}$ and $\{x_8\}$, and $\{x_1, x_2, x_3\}$ is further divided into $\{x_1\}$ and $\{x_2, x_3\}$. This results in five clusters: $\{x_1\}$, $\{x_2, x_3\}$, $\{x_8\}$, $\{x_4, x_5\}$ and $\{x_6, x_7\}$.

The dendrogram in Fig. 7.5 illustrates these clusters, as well as the partitions with 6, 7 and 8 clusters at successive levels. The same is represented in an onion layer diagram (Fig. 7.6), which shows the cluster generation order numerically. Observe that at the final level, each cluster has only one point; such clusters are called singleton clusters.

It is important to highlight that in order to find the optimal 2-partition for a cluster of size m , it is necessary to consider $2^m - 1$ possible 2-partitions and select the best one. Therefore, generating

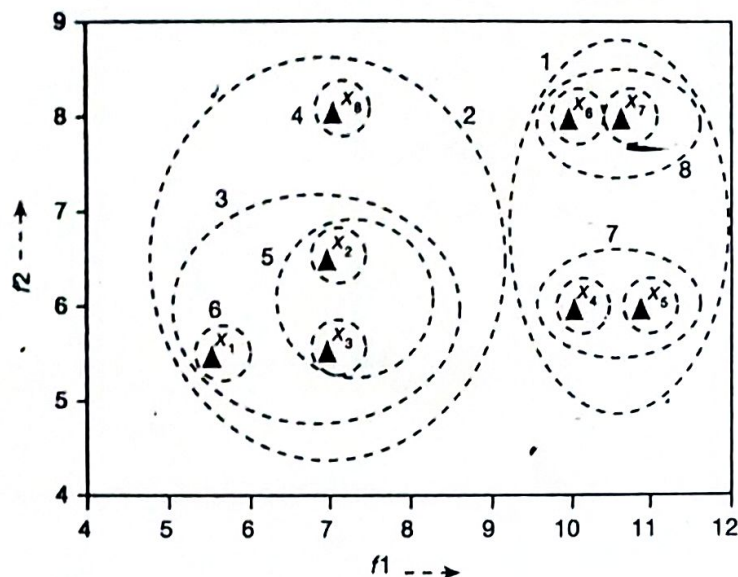


FIG. 7.6 Divisive clustering with cluster generation order for the data points listed in Table 7.12

all possible 2-partitions and choosing the most suitable partition requires an effort of $\mathcal{O}(2^n)$, where n represents the number of data points.

Monothetic clustering involves considering each feature direction individually and dividing the data into two clusters based on the gap in projected values along that feature direction. Specifically, the data set is split into two parts at a point that corresponds to the mean value of the maximum gap observed among the feature values.

This process is then repeated sequentially for the remaining features, further partitioning each cluster. The following example illustrates monothetic clustering

EXAMPLE 12: Consider the eight data points provided in Table 7.12, each having two features, f_1 and f_2 . The first step is to split the data based on the maximum inter-pattern gap observed in the f_1 direction. Sorting the f_1 values in ascending order yields $x_1 : 5.5$, $x_2 : 7$, $x_3 : 7$, $x_8 : 7$, $x_4 : 10$, $x_6 : 10$, $x_7 : 10.5$ and $x_5 : 10.75$. The largest gap of 3 units is between x_8 and x_4 . We select the mid-point between 7 and 10, which is 8.5 (that is, $7 + 1.5$), and use it to split the data into two clusters: $C1 = \{x_1, x_2, x_3, x_8\}$ and $C2 = \{x_4, x_5, x_6, x_7\}$.

Next, each of these clusters is further divided based on the f_2 values. Sorting the patterns in $C1$ by their f_2 values gives $x_1 : 5.5$, $x_3 : 5.5$, $x_2 : 6.5$ and $x_8 : 8$. The largest gap of 1.5 units ($8 - 6.5 = 1.5$ units) occurs between x_2 and x_8 . We split $C1$ at the midpoint 7.25 (that is, $6.5 + 0.75$) along the f_2 direction, resulting in two clusters: $C11 = \{x_8\}$ and $C12 = \{x_1, x_2, x_3\}$. Similarly, by splitting $C2$ using the value 7 for f_2 , we obtain $C21 = \{x_6, x_7\}$ and $C22 = \{x_4, x_5\}$; by splitting $C12$ using the value 6.25 for f_1 , we obtain $C121 = \{x_1\}$ and $C122 = \{x_2, x_3\}$. Figure 7.7 shows this monothetic clustering.

However, a challenge with this approach is that in the worst case, the initial data set may be divided into 2^l clusters when considering all l features. Having such a large number of clusters might not be practical, necessitating an additional merging phase. In this phase, pairs of clusters are selected based on their proximity and merged into a single cluster. This process is repeated until the desired number of clusters is achieved.

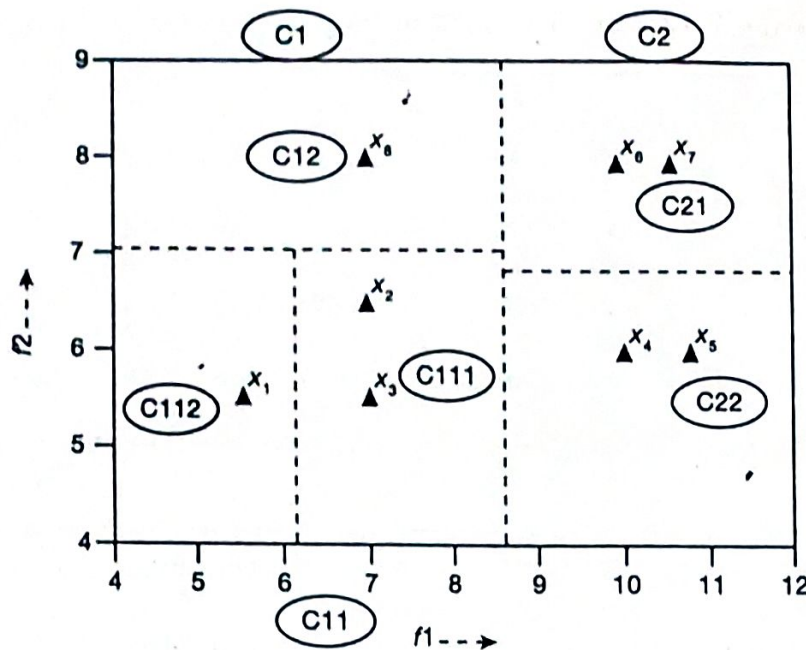


FIG. 7.7 Monothetic clustering for the data points listed in Table 7.12

One commonly used criterion for proximity is the distance between the centroids of the clusters. The time complexity of sorting the n elements and finding the maximum inter-pattern gap in each feature direction is $O(n \log n)$. Considering l features, the overall effort is $O(ln \log n)$. However, this approach may become infeasible for large values of l and n .

7.4 AGGLOMERATIVE CLUSTERING

An agglomerative clustering algorithm generally uses the following steps:

1. Compute the proximity matrix for all pairs of patterns in the data set.
2. Find the closest pair of clusters based on the computed proximity measure and merge them into a single cluster. Update the proximity matrix to reflect the merge, adjusting the distances between the newly formed cluster and the remaining clusters.
3. If all the patterns belong to a single cluster, terminate the algorithm. Otherwise, go back to Step 2 and repeat the process until all the patterns are in one cluster.

By iteratively merging the closest clusters, the algorithm gradually builds a hierarchy of clusters, with each iteration reducing the number of clusters until a stopping criterion is met. The following example illustrates agglomerative clustering.

EXAMPLE 13: Consider the eight data points specified in Table 7.12. To start with, each data point is a singleton cluster. As per the method, the first step is to compute the proximity between the data points. Table 7.13 shows the proximity between the data points using city-block distance. Note that the city-block (Manhattan) distance between any two data points, x_1 and x_2 , is given by $M(x_1, x_2) = \sum_{i=1}^l |x_1(i) - x_2(i)|$, where l is the number of features in the data point.

TABLE 7.13 City-block distance between pairs of data points

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	0	2.5	1.5	5	5.5	7	7.5	4
x_2	2.5	0	1	3.5	4.25	4.5	5	1.5
x_3	1.5	1	0	3.5	4.25	5.5	6	2.5
x_4	5	3.5	3.5	0	0.75	2	2.5	5
x_5	5.75	4.25	4.25	0.75	0	2.75	2.25	5.75
x_6	7	4.5	5.5	2	2.75	0	0.5	3
x_7	7.5	5	6	2.5	2.25	0.5	0	3.5
x_8	4	1.5	2.5	5	5.75	3	3.5	0

The second step is to find the closest pair of clusters, whose city-block distance is minimum. Since the clusters $\{x_6\}$ and $\{x_7\}$ are the closest to each other with a distance of 0.5 units, they are merged as C_1 to realise a partition of 7 clusters.

The proximity matrix after the merger is given in Table 7.14. The merging uses a *single-link* strategy. That is, the distance between any pair of clusters C_p and C_q is $\min_{x_i \in C_p \text{ and } x_j \in C_q} (d(x_i, x_j))$.

TABLE 7.14 City-block distance among the data points after merging x_6 and x_7 as one cluster

	x_1	x_2	x_3	x_4	x_5	$C_1 = \{x_6, x_7\}$	x_8
x_1	0	2.5	1.5	5	5.5	7	4
x_2	2.5	0	1	3.5	4.25	4.5	1.5
x_3	1.5	1	0	3.5	4.25	5.5	2.5
x_4	5	3.5	3.5	0	0.75	2	5
x_5	5.75	4.25	4.25	0.75	0	2.75	5.75
$C_1 = \{x_6, x_7\}$	7	4.5	5.5	2	2.25	0	3
x_8	4	1.5	2.5	5	5.75	3	0

In the third step, we need to repeat Step 2 till we reach a single cluster of the required number of clusters. In Table 7.14, one can observe that the pair of clusters $\{x_4\}$ and $\{x_5\}$ has a minimum distance (that is, 0.75 units) and hence the clusters are merged next to form $C_2 = \{x_4, x_5\}$. Similarly, the clusters $\{x_2\}$ and $\{x_3\}$ are merged to create $C_3 = \{x_2, x_3\}$. Then, C_3 is merged with cluster $\{x_1\}$, resulting in $C_4 = \{x_1, x_2, x_3\}$. Subsequently, C_4 is merged with cluster $\{x_8\}$, leading to $C_5 = \{x_1, x_2, x_3, x_8\}$. Clusters C_2 and C_1 are then merged to form $C_6 = \{x_4, x_5, x_6, x_7\}$.

At this point, there are two clusters remaining, namely, C_5 and C_6 . The process can be terminated once the desired number of clusters is achieved. The dendrogram in Fig. 7.8 illustrates the merging of clusters at different levels.

The time complexity of agglomerative clustering is $O(n^2)$, where n is the number of data points. This is to compute the proximity between all pairs of data points in the data set. Since the computed proximity matrix needs to be stored, the space complexity is also $O(n^2)$.

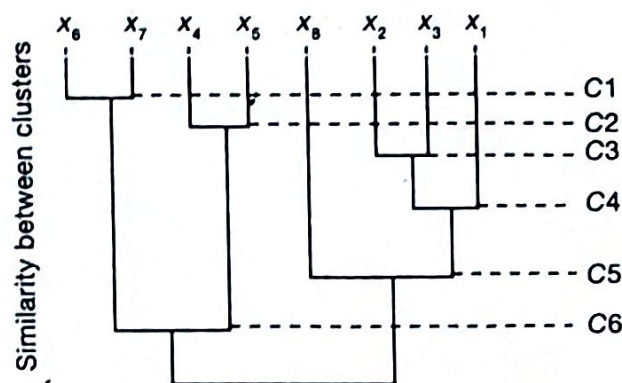


FIG. 7.8 Agglomerative clustering for the data points listed in Table 7.12

7.5 PARTITIONAL CLUSTERING

We have discussed a partitional clustering algorithm, the Leader clustering algorithm, in an earlier chapter. One of the most popular algorithms under this category is the k -means clustering algorithm.

7.5.1 K -Means Clustering

The k -means algorithm can be described in the following steps:

1. Select k initial cluster centres from the given n data points. The remaining $(n - k)$ data points are then assigned to one of the k clusters based on their proximity to the closest cluster centre.
2. Compute the new cluster centres based on the current assignment of data points. This is done by calculating the mean or centroid of all the data points belonging to each cluster.
3. Assign each of the n data points to the cluster centre that is closest in proximity to it.
4. Check if there have been any changes in the assignment of data points to clusters. If no changes have occurred, terminate the algorithm; otherwise, go back to Step 2 and repeat the process.

By iteratively updating the cluster centres and reassigning data points until convergence, the k -means algorithm aims to find a partition that minimizes the overall within-cluster variance or distance.

We illustrate the algorithm using the following example.

EXAMPLE 14: Consider the data set containing eight points with two features as shown in Table 7.15.

Let us assume that the number of clusters, $k = 3$. If we select the initial clusters as x_1 , x_4 and x_7 , then Cluster 1 has $(1, 1)$ as its cluster centre, Cluster 2 has $(7, 2)$ as its cluster centre and Cluster 3 has $(7, 9)$ as its cluster centre.

Table 7.16 shows the Euclidean distance (a proximity measure) between the cluster centre and the other data points. The patterns x_2 , x_3 , x_5 , x_6 and x_8 are assigned to their respective clusters as shown in the last column of Table 7.16.

TABLE 7.15 Example of data patterns with two features

Data points	f1	f2
x_1	1	1
x_2	1	3
x_3	3	3
x_4	7	2
x_5	8	2
x_6	9	3
x_7	7	9
x_8	8	9

TABLE 7.16 Euclidean distances with cluster assignments

Data points	Euclidean distance from C1	Euclidean distance from C2	Euclidean distance from C3	Assigned cluster
x_1	0	$\sqrt{37}$	10	C1
x_2	2	$\sqrt{37}$	$\sqrt{72}$	C1
x_3	$\sqrt{8}$	$\sqrt{17}$	$\sqrt{52}$	C1
x_4	$\sqrt{37}$	0	7	C2
x_5	$\sqrt{50}$	1	$\sqrt{50}$	C2
x_6	$\sqrt{68}$	$\sqrt{5}$	$\sqrt{40}$	C2
x_7	10	7	0	C3
x_8	$\sqrt{113}$	$\sqrt{50}$	1	C3

The next step involves computing the new cluster centres. The new cluster centre of C1 will be the mean of the patterns in Cluster 1 (mean of x_1 , x_2 and x_3), which will be (1.67, 2.33). The cluster centre of C2 will be (8, 2.33) and the cluster centre of C3 will be (7.5, 9).

The patterns are again assigned to the closest cluster depending on the distance from the cluster centres. Now, x_1 , x_2 and x_3 are assigned to Cluster 1, x_4 , x_5 and x_6 are assigned to Cluster 2 and x_7 and x_8 are assigned to Cluster 3. Table 7.17 shows the Euclidean distance from the new cluster centre to the data points with their new assigned clusters.

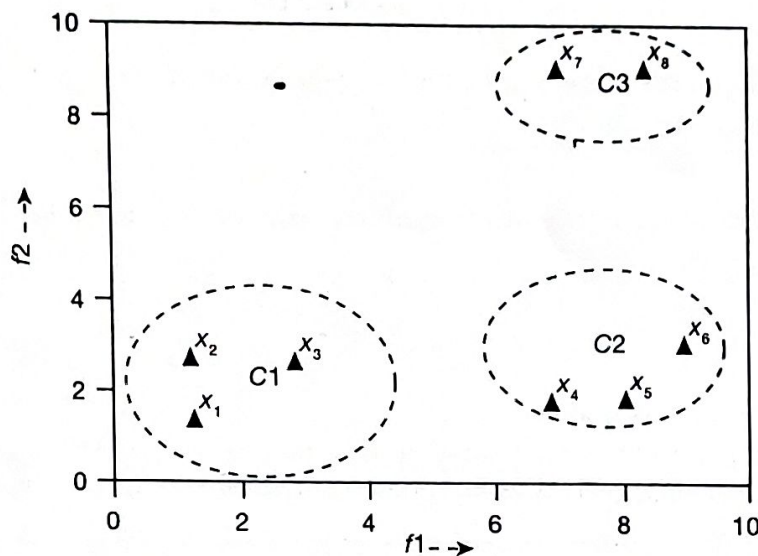
Since there is no change in the clusters formed, this is the final set of clusters. The visual representation of the clusters is given in Fig. 7.9.

One of the important points to be noted in k -means clustering is that the algorithm is *sensitive to the selection of initial centroids*. That is, cluster formation changes with the initially chosen cluster centres. For example, if we select the initial cluster centre as x_1 , x_2 and x_3 , the clustering generated is as shown in Fig. 7.10. (Cluster formation using k -means clustering with $k = 3$ is left as an exercise to the reader.)

The goodness of the k -means clustering algorithm is based on the sum of the squared deviations of data points in a cluster from its centre. More formally, if C_i is the i^{th} cluster and μ_i is its centre,

TABLE 7.17 Euclidean distances with cluster assignments

Data points	Euclidean Euclidean from C1	Euclidean distance from C2	Euclidean distance from C3	Assigned cluster
x_1	1.49	7.13	10.3	C1
x_2	0.95	7	8.85	C1
x_3	1.49	5.24	7.5	C1
x_4	5.34	1.05	7.02	C2
x_5	6.34	0.33	7.02	C2
x_6	7.36	1.2	6.18	C2
x_7	8.54	6.74	0.5	C3
x_8	9.2	6.67	0.5	C3

FIG. 7.9 *K*-means clustering (with $k = 3$) for the data points listed in Table 7.15

then the criterion function minimised by the algorithm is

$$\sum_{i=1}^k \sum_{x \in C_i} (x - \mu_i)^t (x - \mu_i)$$

This is called the **sum-of-squared-error criterion**. This should be minimal for optimal clustering.

Choosing a suitable value for k is a practical challenge in k -means clustering. A commonly employed strategy is to initialize the cluster centres for k in a way that maximizes their distance from each other, which has shown to be effective in real-world scenarios. Alternatively, the **elbow method** is a popular approach to determine the value of k . The steps used in the elbow method are as follows:

1. Perform the k -means clustering algorithm (as discussed earlier) for different values of k .

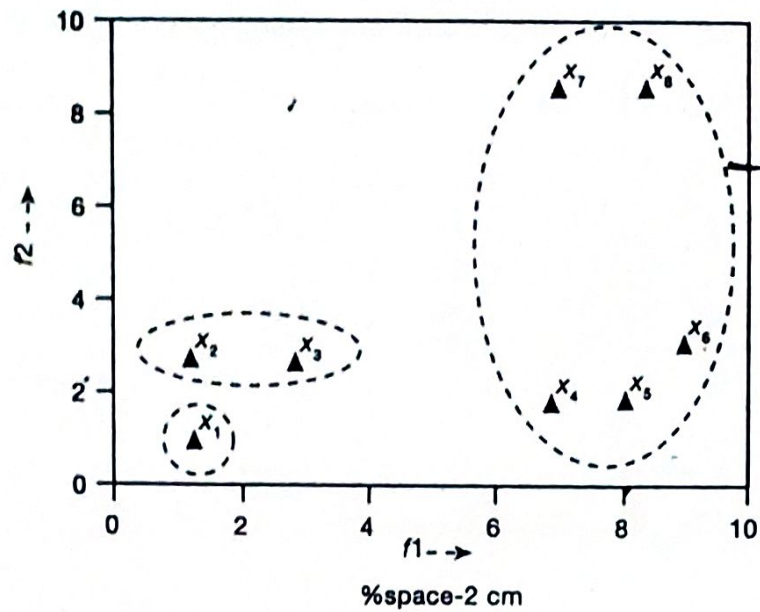


FIG. 7.10 K -means clustering (with $k = 3$) for the data points listed in Table 7.15 but with initial cluster centre, x_1, x_2 and x_3

- For each k , calculate the total in-cluster sum of squared error (IC-SSE):

$$\text{IC-SSE} = \sum_{i=1}^k \sum_{x \in C_i} (x - \mu_i)^2$$

- Plot the curve, IC-SSE versus k .
- The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate value of k . Figure 7.11 is an indicative plot showing the elbow method.

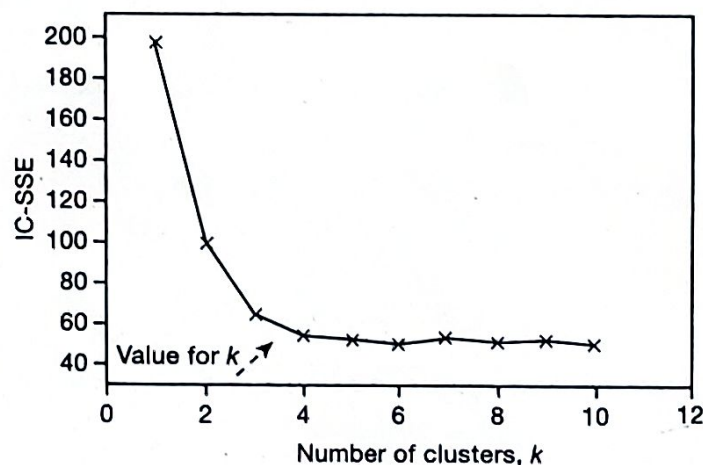


FIG. 7.11 Elbow method for approximating the k value

The k -means clustering algorithm has a time complexity of $\mathcal{O}(nlkp)$, where n represents the number of data points, l denotes the number of features in each data point, k specifies the number of clusters and p indicates the number of iterations. As for space requirements, it is $\mathcal{O}(kn)$. These

characteristics of the algorithm make it more practical and hence it is considered as one of the popular clustering algorithms.

7.5.2 K-Means++ Clustering

In the case of k -means clustering, we randomly select the initial k cluster centres (centroids). We know that in k -means clustering, cluster formation depends on the initial selection of the centroids. To obtain good clusters, we need to select the initial cluster centroids smartly. The k -means++ clustering algorithm is mainly used for identifying the initial cluster centres. The idea here is to choose the initial k centroids away from each other, in a probabilistic sense, so that they become good representatives of the clusters formed.

The steps to identify the initial k centroids are as follows:

1. Initially, select a random data point as the first centroid.
2. Compute the distance (say Euclidean distance) between each centroid and the other data points. Record the minimum distances between any centroid and the data point.
3. Compute the probabilities among the computed minimum distances.
4. Choose the next centroid with maximum probability among all probabilities computed in Step 3.
5. Repeat steps 2 to 4 till we get k number of centroids.

We illustrate the k -means++ clustering for initialization using the following example.

EXAMPLE 15: Consider the data set with following data points:

$$x_1 = (8, 5), x_2 = (9, 4), x_3 = (6, 10)$$

$$x_4 = (4, 4), x_5 = (2, 4), x_6 = (11, 2)$$

Figure 7.12 gives the visual representation of the data set.

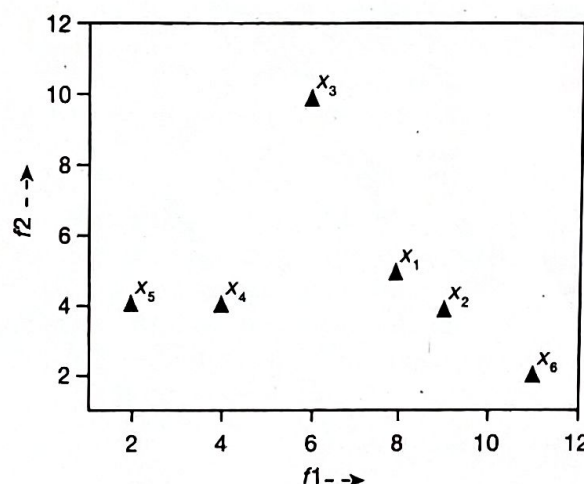


FIG. 7.12 Visualization of the data set

Let x_1 be the randomly selected first cluster centre. Table 7.18 gives the Euclidean distances and probabilities between x_1 and the other data points. Choose x_5 as the second centroid, as its probability in the list is maximum.

TABLE 7.18 Euclidean distances and probabilities between x_1 and other data points

Data point	$d(x_1, x_i)$	$\text{Min}(d(x_1, x_i))$	$\text{Prob}_i = \frac{d(x_1, x_i)}{\sum d(x_1, x_i)}$
x_1	-	-	-
x_2	1.41	1.41	0.066
x_3	5.39	5.39	0.254
x_4	4.12	4.12	0.194
x_5	6.08	6.08	0.286
x_6	4.24	4.24	0.2

Table 7.19 gives the Euclidean distances and probabilities between x_1, x_5 and other data points

TABLE 7.19 Euclidean distances and probabilities between x_1, x_5 and other data points

Data point	$d(x_1, x_i), d(x_5, x_i)$	$\text{Min}(d(x_1, x_i), d(x_5, x_i))$	$\text{Prob}_i = \frac{\text{Min}(d(x_1, x_i), d(x_5, x_i))}{\sum \text{Min}(d(x_1, x_i), d(x_5, x_i))}$
x_1	-	-	-
x_2	1.41, 7.0	1.41	0.09
x_3	5.39, 7.2	5.39	0.36
x_4	4.12, 2	4.12	0.27
x_5	-	-	-
x_6	4.24, 9.22	4.24	0.28

Choose x_3 as the third centroid, as its probability in the list is maximum. We now run the k -means clustering algorithm with initial centroids, x_1, x_2 and x_3 .

Note that in k -means++, we have some initialization (set-up) cost when compared to the conventional k -means algorithm. But convergence tends to be faster and better with lower heterogeneity. One can visualize this in Fig. 7.13.

Figure 7.13 represents cluster formation using random initial centroids and selection of initial centroids by k -means++. Figure 7.13 (a) shows the randomly selected initial three centroids (circled), Fig. 7.13 (b) shows three possible clusters after executing the k -means algorithm. Figure 7.13 (c) shows the initial 3-centroids (circled) by k -means++ and Fig. 7.13 (d) shows cluster formation after executing the k -means clustering algorithm.

Visually it is clear that (i) the amount of centroid movement is less if we use the k -means++ initialization step (the grey arrows in both the cases shows this) and (ii) the cluster formed has lower heterogeneity with k -means++ initialization.

7.5.3 Soft Partitioning

The conventional k -means algorithm assigns data points exclusively to the cluster which is nearest to it, and this leads to a hard partition. Due to inadequate selection of the initial cluster centre, the algorithm may result in a locally optimal solution. Various solutions were proposed in the literature to address this issue.

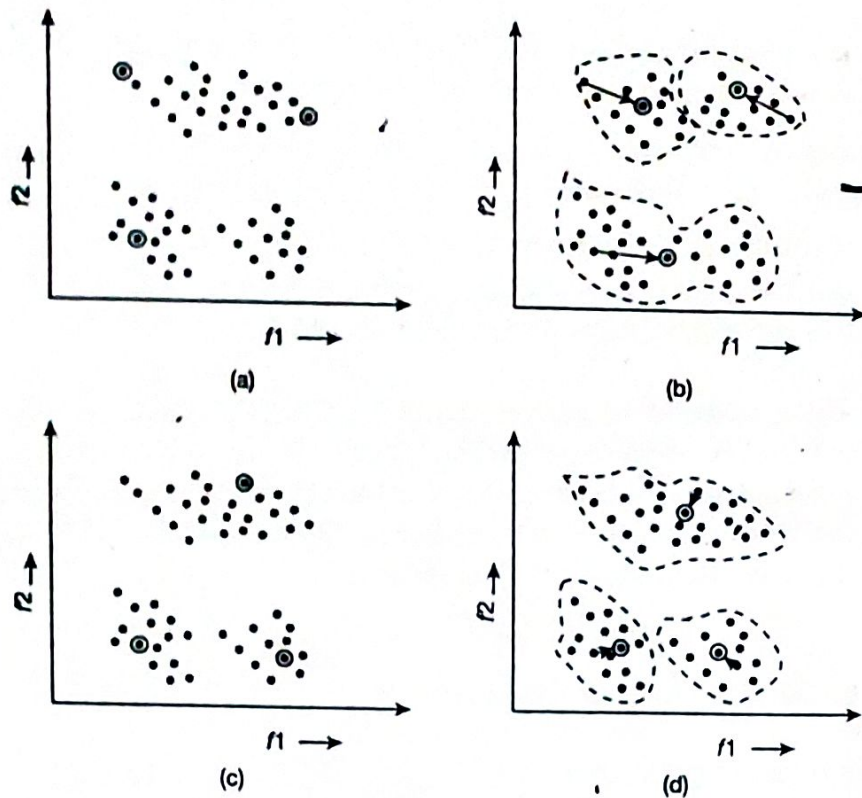


FIG. 7.13 Visualization of goodness of using k -means clustering for initialization (for colour figure, please see Colour Plate 2)

Split and merge strategy: Here, clusters generated by the k -means algorithm are evaluated for potential splitting and merging.

If the sample variance of a cluster exceeds a specified threshold value (τ_v), the cluster is divided into two clusters by selecting the most dissimilar pair of patterns as initial seeds. Likewise, if the distance between the centres of two clusters is below a distance threshold (τ_d), they are merged into a single cluster. By appropriately setting τ_v and τ_d , it becomes possible to obtain an optimal partition. For example, the cluster $\{x_4, x_5, x_6, x_7, x_8\}$ in Fig. 7.10 is selected for splitting based on the variance surpassing the user-defined threshold, τ_v . By choosing either x_4, x_5 or x_6 as initial centroids for one cluster and either x_7 or x_8 for the other, we obtain the clusters $\{x_4, x_5, x_6\}$ and $\{x_7, x_8\}$.

Additionally, by merging the clusters $\{x_1\}$ and $\{x_2, x_3\}$ because the distance between x_1 and the centroid of $\{x_2, x_3\}$ falls below the user-defined threshold τ_d , the partition shown in Fig. 7.9 can be achieved. However, implementing this strategy is challenging due to the difficulty of estimating τ_v and τ_d .

Multiple membership approach: This approach involves the influence of not only the closest cluster centre to a given data point but also neighboring cluster centres. These cluster centroids are affected to varying degrees by the data point, with the closest centre being more influenced than others. Several existing methods for this approach include:

- **Fuzzy clustering:** Each data point is assigned to multiple clusters, typically more than one, based on a membership value. The value is computed using the data point and the corresponding cluster centroid.
- **Rough clustering:** Each cluster is assumed to have both a non-overlapping part and an overlapping part. Data points in the non-overlapping portion belong exclusively to that cluster, while data points in the overlapping part may belong to multiple clusters.

- **Neural network-based clustering:** In this method, varying weights associated with the data points are used to obtain a soft partition.

Stochastic approach: In this category, the focus is on achieving the global optimal solution for cluster formation through probabilistic methods. Some existing methods for this approach include:

- **Simulated annealing:** In this case, the current solution is randomly updated, and the resulting solution is accepted with a certain probability. If the resulting solution is better than the current solution, it is accepted; otherwise, it is accepted with a probability ranging from 0 to 1.
- **Tabu search:** Unlike simulated annealing, multiple solutions are stored, and the current solution is modified in various ways to determine the next configuration.
- **Evolutionary algorithms:** This method maintains a population of solutions. In addition to the fitness values of individuals, a random search based on the interaction among solutions with mutation is employed to generate the next population.

7.6 SOFT CLUSTERING

In numerous applications, assigning an object or pattern (data point) to a single cluster is often not feasible.

For instance, when assigning a document in text mining, it is possible for the same document to fall under both the “politics” and “sports” categories. Similarly, a person in a social network can belong to multiple subject areas within social media. This means that an object or pattern can be assigned to more than one category. This type of categorization, where an object or pattern can be associated with multiple groups, is known as soft partitioning or soft clustering.

The general form of soft clustering involves considering n data points, denoted as x_1, x_2, \dots, x_n , and k clusters, denoted as C_1, C_2, \dots, C_k . In soft clustering, a data point can belong to one or more clusters. This relationship can be represented by an $n \times k$ matrix, where a non-zero entry in the cell (i, j) indicates that the data point x_i belongs to the cluster C_j , while $(i, j) = 0$ indicates otherwise.

In each row, the number of non-zero entries can range from 1 to k . Excluding the all-zero vector, which implies that x_i must be assigned to at least one of the k clusters, there are $(2^k - 1)$ possible choices for each row in terms of the non-zero entries. Therefore, the total number of possibilities for all n rows is $(2^k - 1)^n$. If we store one of p possible values in each cell to indicate the extent to which x_i belongs to C_j , then the number of possibilities is bounded by $(p^k - 1)^n$, considering the number of distinct values each cell can assume as p .

7.6.1 Fuzzy C-Means Clustering

In this type, each data point x_i can be assigned to a cluster C_j with a membership value μ_{ij} , which represents the degree to which x_i belongs to C_j . The membership value μ_{ij} is assumed to range between 0 and 1, and for each data point x_i , the sum of all μ_{ij} values across clusters is equal to 1.

The fuzzy c -means clustering algorithm follows an iterative scheme similar to the k -means algorithm. It begins by identifying c initial cluster centres from the data set or membership values between each data point and the cluster. The following steps are then performed iteratively:

1. Compute membership values:

$$\mu_{ij} = \frac{d(x_j, C_i)^{-1/(M-1)}}{\sum_{l=1}^c d(x_j, C_l)^{-1/(M-1)}}$$

2. Compute the fuzzy cluster centroid:

$$C_i = \frac{\sum_{j=1}^m (\mu_{ij})^M \times x_j}{\sum_{j=1}^m (\mu_{ij})^M}$$

In the above expressions, μ_{ij} represents the membership value of data point x_j in cluster C_i , where $j = 1, 2, \dots, n$ and $i = 1, 2, \dots, c$; $d(x_j, C_i)$ denotes the Euclidean distance between x_j and C_i ; C_i represents the centroid of the i th cluster and M is the fuzzyifying constant which determines the behaviour of the algorithm. When $M = 1$, the fuzzy algorithm functions similar to the hard k -means algorithm, where $\mu_{ij} = 1$ when X_j is assigned to C_i . As M increases or tends to infinity, μ_{ij} approaches $\frac{1}{c}$, as the exponents in both the numerator and denominator tend towards 0.

3. Repeat the above steps, and the algorithm terminates when there is no significant change in the computed values for membership and cluster centroid.

EXAMPLE 16: Consider the data points with two features shown in Table 7.21.

TABLE 7.20 Example of a data set with two features

Data points	f1	f2
x_1	1	1
x_2	2	2
x_3	4	3
x_4	5	3

Let the number of clusters, $c = 2$ and $M = 2$. To start with, assume the following membership values between the data points and the clusters ($\mu^{(0)}$):

	x_1	x_2	x_3	x_4
C1	1	1	0	0
C2	0	0	1	1

The cluster centres are calculated as follows:

$$C1_{f1} = \frac{\mu_{11}^2 * x_1(f1) + \mu_{12}^2 * x_2(f1) + \mu_{13}^2 * x_3(f1) + \mu_{14}^2 * x_4(f1)}{\mu_{11}^2 + \mu_{12}^2 + \mu_{13}^2 + \mu_{14}^2}$$

$$C1(f1) = \frac{1^2 * 1 + 1^2 * 2 + 0^2 * 4 + 0^2 * 5}{1^2 + 1^2 + 0^2 + 0^2}$$

$$C1(f1) = \frac{3}{2} = 1.5$$

$$C1(f2) = \frac{\mu_{11}^2 * X1_{f2} + \mu_{12}^2 * X2_{f2} + \mu_{13}^2 * X3_{f2} + \mu_{14}^2 * X4_{f2}}{\mu_{11}^2 + \mu_{12}^2 + \mu_{13}^2 + \mu_{14}^2}$$

$$C1(f2) = \frac{1^2 * 1 + 1^2 * 2 + 0^2 * 3 + 0^2 * 3}{1^2 + 1^2 + 0^2 + 0^2}$$

$$C1(f2) = \frac{3}{2} = 1.5$$

So, the updated cluster centre for $C1 = (1.5, 1.5)$.

Similarly,

$$C2(f1) = \frac{\mu_{21}^2 * x_1(f1) + \mu_{22}^2 * x_2(f1) + \mu_{23}^2 * x_3(f1) + \mu_{24}^2 * x_4(f1)}{\mu_{21}^2 + \mu_{22}^2 + \mu_{23}^2 + \mu_{24}^2}$$

$$C2(f1) = \frac{0^2 * 1 + 0^2 * 2 + 1^2 * 4 + 1^2 * 5}{0^2 + 0^2 + 1^2 + 1^2}$$

$$C2(f1) = \frac{9}{2} = 4.5$$

$$C2(f2) = \frac{\mu_{21}^2 * X1_{f2} + \mu_{22}^2 * X2_{f2} + \mu_{23}^2 * X3_{f2} + \mu_{24}^2 * X4_{f2}}{\mu_{21}^2 + \mu_{22}^2 + \mu_{23}^2 + \mu_{24}^2}$$

$$C2(f2) = \frac{0^2 * 1 + 0^2 * 2 + 1^2 * 3 + 1^2 * 3}{0^2 + 0^2 + 1^2 + 1^2}$$

$$C2(f2) = \frac{6}{2} = 3$$

So, the updated cluster centre for $C2 = (4.5, 3)$.

The next step is recalculation of membership values. As a part of this step, let us first calculate the Euclidean distance between the cluster centres and the data points. This is done as follows:

$$d(C1, x_1) = d((1.5, 1.5), (1, 1)) = \sqrt{1.5 - 1)^2 + (1.5 - 1)^2} = 0.71$$

$$d(C1, x_2) = d((1.5, 1.5), (2, 2)) = \sqrt{(1.5 - 2)^2 + (1.5 - 2)^2} = 0.71$$

$$d(C1, x_3) = d((1.5, 1.5), (4, 3)) = \sqrt{(1.5 - 4)^2 + (1.5 - 3)^2} = 2.92$$

$$d(C1, x_4) = d((1.5, 1.5), (5, 3)) = \sqrt{(1.5 - 5)^2 + (1.5 - 3)^2} = 3.81$$

$$d(C2, x_1) = d((4.5, 3), (1, 1)) = \sqrt{(4.5 - 1)^2 + (3 - 1)^2} = 4.03$$

$$d(C2, x_2) = d((4.5, 3), (2, 2)) = \sqrt{(4.5 - 2)^2 + (3 - 2)^2} = 2.69$$

$$d(C2, x_3) = d((4.5, 3), (4, 3)) = \sqrt{(4.5 - 4)^2 + (3 - 3)^2} = 0.25$$

$$d(C2, x_4) = d((4.5, 3), (5, 3)) = \sqrt{(4.5 - 5)^2 + (3 - 3)^2} = 0.25$$

The membership values are:

$$\mu_{11} = \frac{\frac{1}{d(C1, x_1)}}{\frac{1}{d(C1, x_1)} + \frac{1}{d(C2, x_1)}}$$

$$\mu_{11} = \frac{\frac{1}{0.71}}{\frac{1}{0.71} + \frac{1}{4.03}} = 0.85$$

$$\mu_{12} = \frac{\frac{1}{d(C1, x_2)}}{\frac{1}{d(C1, x_2)} + \frac{1}{d(C2, x_2)}}$$

$$\mu_{12} = \frac{\frac{1}{0.71}}{\frac{1}{0.71} + \frac{1}{2.69}} = 0.79$$

$$\mu_{13} = \frac{\frac{1}{d(C1, x_3)}}{\frac{1}{d(C1, x_3)} + \frac{1}{d(C2, x_3)}}$$

$$\mu_{13} = \frac{\frac{1}{2.92}}{\frac{1}{2.92} + \frac{1}{0.25}} = 0.08$$

$$\mu_{14} = \frac{\frac{1}{d(C1, x_4)}}{\frac{1}{d(C1, x_4)} + \frac{1}{d(C2, x_4)}}$$

$$\mu_{14} = \frac{\frac{1}{3.81}}{\frac{1}{3.81} + \frac{1}{0.25}} = 0.06$$

Similarly, μ_{21} , μ_{22} , μ_{23} , μ_{24} can be computed and their values are 0.15, 0.21, 0.92 and 0.94, respectively. Note that for a given data point x_j , $\sum_{i=1}^2 (\mu_{ij}) = 1$.

Now the updated membership values $\mu^{(1)}$ are given below:

	x_1	x_2	x_3	x_4
C1	0.85	0.79	0.08	0.06
C2	0.15	0.21	0.92	0.94

Since we do not end up with the same cluster centres or the same membership values, we iterate for the next step. [Note: To converge in a finite number of iterations, some threshold values, either for the difference between the current and previous membership values or for the difference between the current and the previous cluster centre values, can be specified.]

The updated cluster centres for the updated membership values ($\mu^{(1)}$) are as follows:

$$C1(f1) = \frac{0.85^2 * 1 + 0.79^2 * 2 + 0.08^2 * 4 + 0.06^2 * 5}{0.85^2 + 0.79^2 + 0.08^2 + 0.06^2}$$

$$C1(f1) = \frac{2.04}{1.36} = 1.5$$

$$C1_{f2} = \frac{0.85^2 * 1 + 0.79^2 * 2 + 0.08^2 * 3 + 0.06^2 * 3}{0.85^2 + 0.79^2 + 0.08^2 + 0.06^2}$$

$$C1(f2) = \frac{2.01}{1.36} = 1.48$$

So, the updated cluster centre for $C1 = (1.5, 1.48)$.

$$C2(f1) = \frac{0.15^2 * 1 + 0.21^2 * 2 + 0.92^2 * 4 + 0.94^2 * 5}{0.15^2 + 0.21^2 + 0.92^2 + 0.94^2}$$

$$C2(f1) = \frac{7.9143}{1.797} = 4.4$$

$$C2(f2) = \frac{0.15^2 * 1 + 0.21^2 * 2 + 0.92^2 * 3 + 0.94^2 * 3}{0.15^2 + 0.21^2 + 0.92^2 + 0.94^2}$$

$$C2(f2) = \frac{5.307}{1.7966} = 2.95$$

So, the updated cluster centre for $C2 = (4.4, 2.95)$.

The next step is recalculation of membership values. As a part of this step, let us first calculate the Euclidean distance between the cluster centres and the data points. This is done as follows:

$$d(C1, x_1) = d((1.5, 1.48), (1, 1)) = \sqrt{(1.5 - 1)^2 + (1.48 - 1)^2} = 0.693$$

$$d(C1, x_2) = d((1.5, 1.48), (2, 2)) = \sqrt{(1.5 - 2)^2 + (1.48 - 2)^2} = 0.721$$

$$d(C1, x_3) = d((1.5, 1.48), (4, 3)) = \sqrt{(1.5 - 4)^2 + (1.48 - 3)^2} = 2.925$$

$$d(C1, x_4) = d((1.5, 1.48), (5, 3)) = \sqrt{(1.5 - 5)^2 + (1.48 - 3)^2} = 3.815$$

$$d(C2, x_1) = d((4.4, 2.95), (1, 1)) = \sqrt{(4.4 - 1)^2 + (2.95 - 1)^2} = 3.919$$

$$d(C2, x_2) = d((4.4, 2.95), (2, 2)) = \sqrt{(4.4 - 2)^2 + (2.95 - 2)^2} = 2.58$$

$$d(C2, x_3) = d((4.4, 2.95), (4, 3)) = \sqrt{(4.4 - 4)^2 + (2.95 - 3)^2} = 0.40$$

$$d(C2, x_4) = d((4.4, 2.95), (5, 3)) = \sqrt{(4.4 - 5)^2 + (2.95 - 3)^2} = 0.60$$

The updated membership values are:

$$\mu_{11} = \frac{\frac{1}{0.693}}{\frac{1}{0.693} + \frac{1}{3.919}} = 0.85$$

$$\mu_{12} = \frac{\frac{1}{0.721}}{\frac{1}{0.721} + \frac{1}{2.58}} = 0.79$$

$$\mu_{13} = \frac{\frac{1}{2.925}}{\frac{1}{2.925} + \frac{1}{0.40}} = 0.12$$

$$\mu_{14} = \frac{\frac{1}{3.815}}{\frac{1}{3.815} + \frac{1}{0.60}} = 0.14$$

Similarly, μ_{21} , μ_{22} , μ_{23} , μ_{24} can be computed and their values are 0.15, 0.21, 0.88 and 0.86, respectively.

Now the updated membership values $\mu^{(2)}$ are given below:

	x_1	x_2	x_3	x_4
C1	0.85	0.79	0.12	0.14
C2	0.15	0.21	0.88	0.86

Since we do not end up with the same cluster point or the same membership values, we iterate for the next step.

The updated cluster centres for the updated membership values ($\mu^{(2)}$) are as follows:

$$C1(f1) = \frac{0.85^2 * 1 + 0.79^2 * 2 + 0.12^2 * 4 + 0.14^2 * 5}{0.85^2 + 0.79^2 + 0.12^2 + 0.14^2}$$

$$C1(f1) = \frac{2.04}{1.355} = 1.5$$

$$C1(f2) = \frac{0.85^2 * 1 + 0.79^2 * 2 + 0.12^2 * 3 + 0.14^2 * 3}{0.85^2 + 0.79^2 + 0.12^2 + 0.14^2}$$

$$C1(f2) = \frac{2.01}{1.355} = 1.48$$

The updated cluster centre for C1 = (1.5, 1.48).

$$C2(f1) = \frac{0.15^2 * 1 + 0.21^2 * 2 + 0.92^2 * 4 + 0.94^2 * 5}{0.15^2 + 0.21^2 + 0.92^2 + 0.94^2}$$

$$C2(f1) = \frac{7.9143}{1.797} = 4.4$$

$$C2(f2) = \frac{0.15^2 * 1 + 0.21^2 * 2 + 0.92^2 * 3 + 0.94^2 * 3}{0.15^2 + 0.21^2 + 0.92^2 + 0.94^2}$$

$$C2(f2) = \frac{5.307}{1.797} = 2.95$$

The updated cluster centre for C2 = (4.4, 2.95).

Since there is no change in the cluster centre, the algorithm stops. The final membership value and cluster allocation for the data points is given in Table 7.21.

TABLE 7.21 Cluster formation using fuzzy *c*-means algorithm

Data point	μ^{C1}	μ^{C2}	Allocated cluster
x_1	0.85	0.15	C1
x_2	0.79	0.21	C1
x_3	0.12	0.88	C2
x_4	0.14	0.86	C2