K-Means Clustering Practical

Specification Requirements

- Python 3.x
- Numpy and/or manual distance calculation logic
- Basic knowledge of Euclidean distance
- Understanding of clustering and centroid computation

Aim

To perform K-means clustering on a given set of 2D points using initial centroids and determine cluster membership, population, and updated centroid positions.

Theory

K-means clustering is an unsupervised learning algorithm that partitions data into K distinct clusters based on distance from centroids.

Steps:

1. Initialize two centroids m1 and m2.
2. Compute the Euclidean distance of each point to the centroids.
3. Assign each point to the nearest centroid to form clusters.
4. Calculate new centroids as the mean of the points in each cluster.
5. Repeat until the centroids do not change significantly or for a fixed number of iterations.

Euclidean distance formula:

d = sqrt((x2 - x1)^2 + (y2 - y1)^2)

Problem Setup

Given Points:

P1 = [0.1, 0.6], P2 = [0.15, 0.71], P3 = [0.08, 0.9], P4 = [0.16, 0.85], P5 = [0.2, 0.3], P6 = [0.25, 0.5], P7 = [0.24, 0.1], P8 = [0.3, 0.2]

Initial Centroids:

m1 = P1 = [0.1, 0.6] (C1), m2 = P8 = [0.3, 0.2] (C2)

Point Assignments

Using Euclidean distance, the assignments are:

P1 -> C1, P2 -> C1, P3 -> C1, P4 -> C1, P5 -> C2, P6 -> C1, P7 -> C2, P8 -> C2

Note- Explain your code in detail

Answers

1. Cluster of P6: Cluster 1 (C1)

2. Population around m2 (C2): 3 (P5, P7, P8)

3. Updated centroids after 1 iteration:

   - New m1 = [0.148, 0.712]

   - New m2 = [0.2467, 0.2]

Conclusion

The k-means clustering algorithm successfully grouped the data into two clusters based on their proximity to the initial centroids. After one iteration, the centroids were updated reflecting the mean positions of the respective clusters. The algorithm can be repeated further to reach stable centroid positions.