13. Demonstrate the use of Fuzzy C-Means Clustering.

Description: **Fuzzy C-Means (FCM)** is an unsupervised clustering algorithm that allows data points to belong to multiple clusters with varying degrees of membership. Unlike **K-Means**, where a point is assigned to exactly one cluster, FCM assigns a probability (membership value) for each point to belong to different clusters.

**Steps in Fuzzy C-Means Clustering:**

1. Initialize cluster centers randomly.

Randomly place C cluster centers in the data space.

1. Compute the membership value of each data point to each cluster.

Each data point is assigned a membership degree **μij** to each cluster based on its distance from the cluster center.

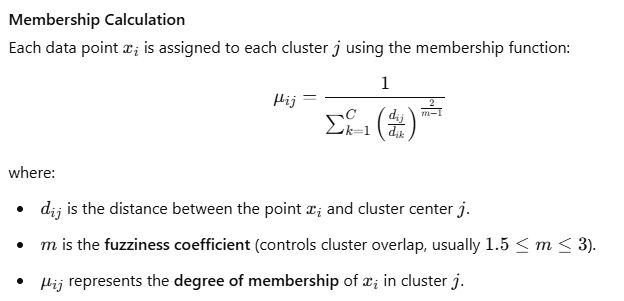
Membership values are **fuzzy probabilities** between 0 and 1.

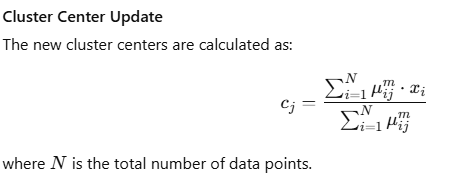
1. Update the cluster centers based on weighted averages.

Weighted averages of data points are used to update cluster centers based on their fuzzy memberships.

1. Repeat steps 2-3 until convergence (i.e., minimal change in cluster centers)

The algorithm stops when cluster centers **stop changing significantly** or after a fixed number of iterations.





**Synthetic Data Generation:**

* 1. Two clusters are generated using np.random.normal().

**Applying FCM:**

* 1. The fuzz.cmeans() function performs Fuzzy C-Means clustering.
  2. The parameter **m=2** controls the fuzziness (higher values allow more overlap between clusters).
  3. The algorithm stops when the **error is below 0.005** or **1000 iterations** are reached.

**Cluster Assignments:**

* 1. The **u matrix** contains membership values for each point.
  2. We assign each point to the cluster with the highest membership probability.

**Visualization:**

* 1. Data points are plotted with their cluster assignments.
  2. Red **"X" markers** indicate cluster centroids.

Installation (If required)

pip install scikit-fuzzy numpy matplotlib

Python Code for FCM

import numpy as np

import skfuzzy as fuzz

import matplotlib.pyplot as plt

# Generate synthetic data (2 clusters)

np.random.seed(42)

cluster1 = np.random.normal(loc=[2, 2], scale=0.5, size=(50, 2))

cluster2 = np.random.normal(loc=[6, 6], scale=0.5, size=(50, 2))

# Combine data into a single dataset

data = np.vstack((cluster1, cluster2)).T # Transpose to match FCM input format

# Number of clusters

num\_clusters = 2

# Apply Fuzzy C-Means

cntr, u, \_, \_, \_, \_, \_ = fuzz.cmeans(data, num\_clusters, m=2, error=0.005, maxiter=1000)

# Get final cluster assignments

cluster\_membership = np.argmax(u, axis=0)

# Plot results

plt.figure(figsize=(8, 6))

for i in range(num\_clusters):

plt.scatter(data[0, cluster\_membership == i], data[1, cluster\_membership == i], label=f'Cluster {i+1}')

plt.scatter(cntr[:, 0], cntr[:, 1], c='red', marker='X', s=200, label='Centroids')

plt.legend()

plt.title('Fuzzy C-Means Clustering')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.show()

Output: