**Problem Statement:**

Apply fuzzy c means algorithm on Boston Housing Dataset.

**Dataset Description:**

The dataset contains 506 samples (rows) with 13 features (columns) and 1 target variable. Each sample represents a particular census tract in the Boston area, with attributes related to housing and demographics. The target variable is the median value of owner-occupied homes in thousands of dollars.

Features (Input Variables):

CRIM: Crime rate per capita (higher value means more crime).

ZN: Proportion of residential land zoned for large lots (in proportion).

INDUS: Proportion of non-retail business acres per town.

CHAS: Charles River dummy variable (1 if tract bounds the river; 0 otherwise).

NOX: Nitrogen oxides concentration (parts per 10 million).

RM: Average number of rooms per dwelling.

AGE: Proportion of owner-occupied units built before 1940.

DIS: Weighted distance to employment centers.

RAD: Index of accessibility to radial highways (higher values mean more access).

TAX: Property tax rate (per $10,000).

PTRATIO: Pupil-teacher ratio by town.

B: Proportion of Black residents by town (calculated as 1000(Bk − 0.63)^2 where Bk is the proportion of Black residents).

LSTAT: Percentage of lower status population.

Target Variable:

MEDV: Median value of homes in thousands of dollars

**Procedure:**

Step 1:

Set the number of clusters (C).

Set the fuzziness parameter (m, usually 2).

Set a convergence criterion or tolerance (epsilon) for when the algorithm will stop.

Step 2: Initialize Membership Matrix:

The sum of membership values for each data point should be 1, i.e.,

Step 3: Initialize Cluster Centers:

Initialize the C cluster centers (centroids) randomly or by selecting random data points as the initial centroids.

These centers represent the central point for each cluster in the feature space.

Step 4: Update Membership Values:

Calculate the degree of membership for each data point in each cluster. The membership values indicate the degree to which each data point belongs to each cluster.

Update the membership matrix using the degree of membership based on the current positions of the centroids and the distances between data points and cluster centers.

Step 5: Update Cluster Centers:

Calculate new cluster centers based on the updated membership values. Each centroid is the weighted average of all data points, where the weights are the membership values raised to a power based on the fuzziness parameter.

Step 6: Check for Convergence:

Check if the algorithm has converged. Convergence is typically determined when:

The change in the membership matrix between iterations is smaller than a predefined threshold (epsilon).

The cluster centers have stabilized (i.e., no significant change in their positions).

Step 7:

If convergence has not been reached, repeat steps 4 through 6. Update the membership matrix and cluster centers iteratively until the algorithm converges or reaches the maximum number of iterations.

Output Results:

Once the algorithm converges, the final cluster centers and membership matrix are obtained.

**Source Code:**

import numpy as np

from sklearn.datasets import fetch\_california\_housing

from sklearn.preprocessing import StandardScaler

import skfuzzy as fuzz

import matplotlib.pyplot as plt

# Load California Housing Dataset

data = fetch\_california\_housing()

X = data.data # Features

# Normalize the data for better clustering performance

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Define the number of clusters

n\_clusters = 3

# Apply Fuzzy C-Means algorithm

cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(

X\_scaled.T, # Transpose to shape (features, samples)

c=n\_clusters, # Number of clusters

m=2.0, # Fuzziness parameter

error=0.005, # Stopping criterion

maxiter=1000, # Maximum number of iterations

init=None # Initial guess for membership values

)

# Assign each data point to the cluster with the highest membership value

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

# Visualize the clustering result (using first two features for simplicity)

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

colors = ['r', 'g', 'b']

for i in range(n\_clusters):

plt.scatter(

X\_scaled[cluster\_membership == i, 0],

X\_scaled[cluster\_membership == i, 1],

label=f'Cluster {i+1}',

color=colors[i],

alpha=0.6

)

plt.title('Fuzzy C-Means Clustering (California Housing Dataset)')

plt.xlabel('Feature 1 (Standardized)')

plt.ylabel('Feature 2 (Standardized)')

plt.legend()

plt.grid()

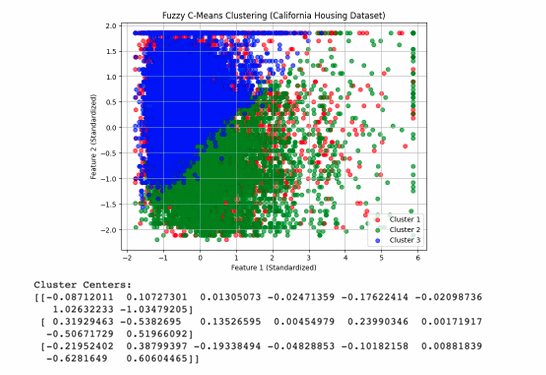
plt.show()

# Print final cluster centers

print("Cluster Centers:")

print(cntr)

**Output**:



**Discussion:**

Fuzzy C-Means (FCM) is a robust clustering algorithm that allows for soft cluster assignments, meaning each data point can belong to multiple clusters with varying degrees of membership. Unlike traditional hard clustering methods like K-means, where each point is assigned to a single cluster, FCM provides a more flexible approach, making it particularly useful for datasets with overlapping or ambiguous cluster boundaries. By iteratively updating the membership values and cluster centers, FCM minimizes an objective function that reflects both the distances between data points and cluster centers and the degree of membership.