Fuzzy c means:

Group members:

-Prateek Vijay(21CE10046)
-pran Kishore mahto(21CE10044)
-pranavish solanki (21CE10045)

Problem statement:

we have m roads and on each road there are n number of vehicles on each road. for each vehicle we have the following attributes: size(2 for 2 wheeler, 4 for 4 wheelers and 8 for trucks/trailers) , their speed on the road . We need to identify the m roads into two clusters , cluster 1 representing the road with less traffic congestion, cluster 2 representing minute congestion and cluster 3 representing highly congested roads. store the membership matrix of the roads.

The code :

```
import numpy as np
from sklearn.datasets import make_blobs
from sklearn.metrics import pairwise_distances_argmin
from sklearn.metrics.pairwise import euclidean_distances

# Define the number of roads and vehicles on each road
m = 10
n = 50

# Generate random data for vehicles on each road,however it can be taken as
input as well
data = np.zeros((m*n, 2))
for i in range(m):
    for j in range(n):
        size = np.random.choice([2, 4, 8])
        speed = np.random.uniform(0, 60)
        data[i*n+j,:] = [size, speed]

# Define the FCM function
def FCM(data, n_clusters=3, max_iter=100, m=2, error=1e-5):
    # Randomly initialize the membership matrix
    membership_mat = np.random.rand(len(data), n_clusters)
    #ensuring the values to be between 0 to 1, we do the following division
    membership_mat = np.divide(membership_mat, np.sum(membership_mat,
axis=1)[:,np.newaxis])

    # Repeat until convergence or max iterations is reached
    iteration = 0
    centroids_final=np.zeroes(n_clusters,2)
    while iteration < max_iter:
        # Compute cluster centers at each iterations
        centroids = np.dot(data.T, membership_mat) / np.sum(membership_mat,
axis=0)

        # Compute distances from data points to cluster centers
        distances = euclidean_distances(data, centroids)

        # Update membership matrix
        membership_mat_new = np.power(distances, -2/(m-1))
        membership_mat_new = np.divide(membership_mat_new,
np.sum(membership_mat_new, axis=1)[:,np.newaxis])

        # Check for convergence
        if np.max(np.abs(membership_mat_new - membership_mat)) < error:
            break

        membership_mat = membership_mat_new
```

```
        centroids_final=centroids
        iteration += 1

    # Assign each road to a cluster based on membership values
    clusters = pairwise_distances_argmin(data, centroids)
    membership_mat = membership_mat[np.arange(len(data)), clusters]

    return clusters, membership_mat,centroids_final

# Compute the sum of vehicle sizes and average speed for each road
road_data = np.zeros((m, 2))
#road_data will be used as the numpy array on which the FCM task will be
performed
for i in range(m):
    #calculating the size of traffic for each road
    road_data[i,0] = np.sum(data[i*n:(i+1)*n,0])
    #calculating the average speed pf the traffic on each road
    road_data[i,1] = np.mean(data[i*n:(i+1)*n,1])

# Apply FCM to cluster the roads based on congestion level
clusters, membership_mat,centroids_final = FCM(road_data)
#print the centroids of each road where the axis =0 represents the size of the traffic
and the axis =1 represents the average speed on that road(here,cluster centre)
for i in centroids_final :
    print (i)
# Print the membership matrix for each road
for i in range(m):
    print("Road {}: Membership: {}".format(i+1, membership_mat[i], ))
```

Explanation:

Here the data points can be grouped into three different clusters .
The idea is to classify the roads and identify the roads with the most traffic congestion.
Through the code, we have found the membership matrix of the road , and the cluster centres. We have the membership values of a road to each clusters. Each cluster has its centroids. Hence we can calculate the value of the following :

- Average speed of traffic on the road:

- Using the values of the road's membership to each cluster, we can calculate the average speed by the summing the product of roads membership to i'th cluster and the average speed on that cluster

-size of traffic :

-we can approximate the size of traffic on the road by summing the product of the road's membership to a cluster and the sizeof that cluster

After the above steps are computed, we can classify the roads as :
-Highly congested.
-mildly congested.
-smooth road.
The above 3 classifications can be done by setting a range of the size of traffic and the average speed of traffic.

Application:

This idea can be implemented at roads with high connectivity with the major cities/districts/ Industrial hubs etc. to ensure smooth flow of traffic. This can be done by having automatic traffic lights, where the traffic lights will go green if the road condition becomes/gets close to highly congested , subsequently stopping the traffic flow on the roads where the traffic is lesser, until the condition gets neutral.

Further applications of the road classification can be done, for example finding the quickest route to a destination, where we can find the size of traffic on the road , and choose our path accordingly.

However , this process might be complicated. This was our idea of implementing the Fuzzy C means into classifying the roads into different traffic congestion state.