Transportation Engineering (Analytical)

Fuzzy C means clustering.

- Pran Kishor Mahto (21CE10044)
- Pranavsinh Solanki (21CE10045)
- Prateek Vijay (21CE10046)

Code & Real life problem example.

Let's suppose we have to find the uses and load or busyness of a particular area or road.

We can analyze it by this clustering and code.

We can assign some particular numerical digit to similar kind of vehicle

Let's take

2: for two wheelers

3: for three wheelers

4: for four wheelers

8: for eight wheeler

We are setting the three speed reader in the area for taking speed of vehicle at different points while vehicle totally cross the area.

Now we have 4 data to analyze the scenario (3 speed data & 1 vehicle type data).

So now we are using four dimensional point data in the code.

Suppose our data is [a,b,c,d] So

here a= speed of vehicle at first

point b= speed of vehicle at second

point c= vehicle type d=speed at

third(exit) point

Now we are taking 2{0,1} clusters.

Cluster 0: it contains slow-moving vehicles

Cluster 1: it contains fast-moving vehicles

Now we are writing a code using some data,

If you want to calculate the same for different data sets, you can change the data in the code itself.

p or m represent the fuzziness (1<p<infinity).

SSE is sum of squared distance of each point from the cluster center.

```
#Fuzzy c means (transportation engineering)
import numpy as np, numpy.random from
scipy.spatial import distance k = 2 p = 5

X = np.array([
    [1,1,2,1],
    [2,1,2,3],
    [2,2,4,5],
    [50,42,2,83],
    [51,43,2,82],
    [51,44,3,89],
    [53,40,8,80]])


# Print the number of data and dimension
n = len(X) d = len(X[0]) addZeros =
np.zeros((n, 1)) X = np.append(X,
addZeros, axis=1) print("The FCM
algorithm: \n") print("The training data:
\n", X) print("\nTotal number of data: ",n)
print("Total number of features: ",d)
print("Total number of Clusters: ",k)

# Create an empty array of centers
C = np.zeros((k,d+1))
#print(C)
```

```
# Randomly initialize the weight matrix weight =
np.random.dirichlet(np.ones(k),size=n)
print("\nThe initial weight: \n", np.round(weight,2))

for it in range(3): # Total number of iterations

   # Compute centroid
   for j in range(k):
      denoSum = sum(np.power(weight[:,j],2))

      sumMM =0
      for i in range(n):
         mm = np.multiply(np.power(weight[i,j],p),X[i,:])
sumMM +=mm        cc = sumMM/denoSum
      C[j] = np.reshape(cc,d+1)

   #print("\nUpdating the fuzzy pseudo partition")
for i in range(n):        denoSumNext = 0
      for j in range(k):            denoSumNext +=
np.power(1/distance.euclidean(C[j,0:d], X[i,0:d]),1/(p-1))
      for j in range(k):
         w = np.power((1/distance.euclidean(C[j,0:d], X[i,0:d])),1/(p-1))/denoSumNext
         weight[i,j] = w

print("\nThe final weights: \n", np.round(weight,2))


for i in range(n):
   cNumber = np.where(weight[i] == np.amax(weight[i]))
   X[i,d] = cNumber[0]
```

```
print("\nThe data with cluster number: \n", X)


SSE = 0 for j in

range(k):     for i in

range(n):

      SSE += np.power(weight[i,j],p)*distance.euclidean(C[j,0:d], X[i,0:d])


print("\nSSE: ",np.round(SSE,4))
```

Now we are attaching the output of this code here.

The training data:

[[1. 1. 2. 1. 0.]

[2. 1. 2. 3. 0.]

[2. 2. 4. 5. 0.]

[50. 42. 2. 83. 0.]

[51. 43. 2. 82. 0.]

[51. 44. 3. 89. 0.]

[53. 40. 8. 80. 0.]]

Total number of data: 7

Total number of features: 4

Total number of Clusters: 2

The initial weight:

[[0.21 0.79]

[0.93 0.07]

[0.27 0.73]

[0.34 0.66]

[0.13 0.87]

[0.98 0.02]

[0.09 0.91]]

The final weights:

[[0.5 0.5]

[0.5 0.5]

[0.5 0.5]

[0.5 0.5]

[0.5 0.5]

[0.5 0.5]

[0.5 0.5]]

The data with cluster number:

[[1. 1. 2. 1. 1.]

[2. 1. 2. 3. 1.]

[2. 2. 4. 5. 1.]

[50. 42. 2. 83. 0.]

[51. 43. 2. 82. 0.]

[51. 44. 3. 89. 0.]

[53. 40. 8. 80. 0.]]

SSE: 25.6931

```
The FCM algorithm:

The training data:
 [[ 1.  1.  2.  1.  0.]
 [ 2.  1.  2.  3.  0.]
 [ 2.  2.  4.  5.  0.]
 [50. 42.  2. 83.  0.]
 [51. 43.  2. 82.  0.]
 [51. 44.  3. 89.  0.]
 [53. 40.  8. 80.  0.]]

Total number of data:  7
Total number of features:  4
Total number of Clusters:  2

The initial weight:
 [[0.21 0.79]
 [0.93 0.07]
 [0.27 0.73]
 [0.34 0.66]
 [0.13 0.87]
 [0.98 0.02]
 [0.09 0.91]]

The final weights:
 [[0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]]

The data with cluster number:
 [[ 1.  1.  2.  1.  1.]
 [ 2.  1.  2.  3.  1.]
 [ 2.  2.  4.  5.  1.]
 [50. 42.  2. 83.  0.]
 [51. 43.  2. 82.  0.]
 [51. 44.  3. 89.  0.]
 [53. 40.  8. 80.  0.]]

SSE:  25.6931
```

Now here we can clearly see in the last output The

slow-moving vehicle is in a different cluster(1).

And the fast moving vehicle is in a different cluster(0).

Thanks