Fuzzy c means clustering

Data $X = \{x_1, x_2, \cdots, x_n\}$, $Clusters = \{C_1, C_2, \cdots, C_k\}$, Where n is the number of data, k is the number of clusters, d is the dimension of the data or number of features/ attributes.

Conditions: (1.) $\sum_{j=1}^{k} w_{ij} = 1$, (2.) $0 < \sum_{i=1}^{n} w_{ij} < n$.

Basic steps of FCM

Input: Data, k

Output: $w_{ij}, C_j, 1 \leq i \leq n, 1 \leq j \leq k$

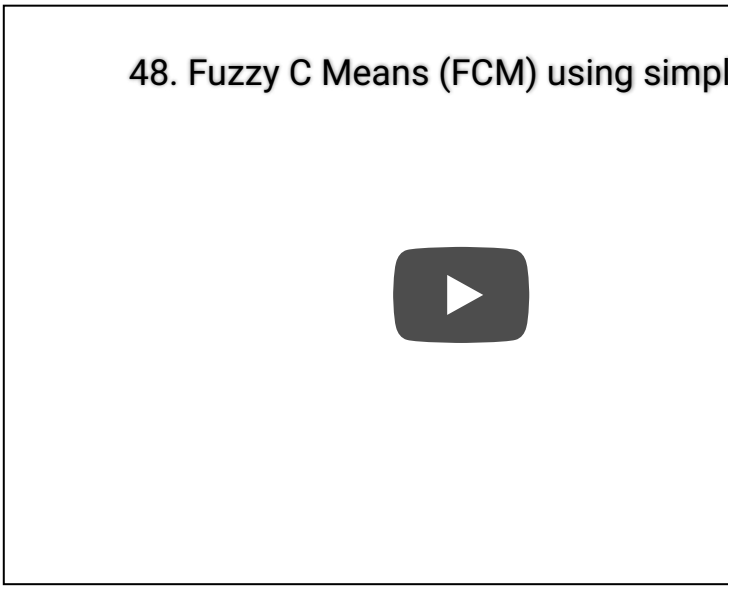
Step 1: w_{ij} values are assigned randomly.

Step 2: (re) calculate centroid of each cluster using the fuzzy-pseudo partition $C_j = \frac{\sum_{i=1}^{n} w_{ij}^p x_i}{\sum_{i=1}^{n} w_{ij}^p}$, p (fuzzy-ness) is 1 to ∞.

Step 3: (re) calculate the fuzzy-pseudo partition $w_{ij} = \frac{(\frac{1}{dist(x_i, c_j)})^{\frac{1}{p-1}}}{\sum_{s=1}^{k} (\frac{1}{dist(x_i, c_s)})^{\frac{1}{p-1}}}$, where $dist(x_i, c_j)$ is the Euclidean distance between x_i data and c_j cluster center.

Step 4: repeat step 2 and 3 if centroids do not change.

For explanation with simple example, you may watch video as follows:



Fuzzy C Means Clustering

...

Python programming implementation of Fuzzy c means clustering algorithm

```
#Fuzzy c means clustering algorithm
import numpy as np, numpy.random
import pandas as pd
from scipy.spatial import distance
k = 2
p = 2

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

```
# To import data from disk; "C:\\MachineLearning
# where the file "fuzzyData.txt" is there
# X = pd.read_csv('C:\\MachineLearning\\Data\\fu
#print(X)


# Print the number of data and dimension
n = len(X)
d = len(X.columns)
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(weigh

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

    # 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,
            sumMM +=mm
        cc = sumMM/denoSum
        C[j] = np.reshape(cc,d+1)

    #print("\nUpdating the fuzzy pseudo partitio
```

```
    for i in range(n):
        denoSumNext = 0
        for j in range(k):
             denoSumNext += np.power(1/distance
        for j in range(k):
            w = np.power((1/distance.euclidean(
            weight[i,j] = w

print("\nThe final weights: \n", np.round(weigh


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

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

# Sum squared error calculation
SSE = 0
for j in range(k):
    for i in range(n):
        SSE += np.power(weight[i,j],p)*distance

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

Leave a Reply

Your email address will not be published.

Required fields are marked *

Comment

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment