FCM 클러스터링 방법

퍼지 군집 (Fuzzy Clustering)은 혼합 분포 군집(Mixture Distribution Clustering)과 함께 "Soft Clustering"이라고도 한다, 각 관측치가 (단 하나의 군집에만 속하는 것이 아니라) 여러 군집에 속할 수 있으며, 이를 각 군집에 속할 가능성(possibility), 확률(probability)로 제시해 준다. 퍼지 군집(Fuzzy Clustering)에서는 퍼지 이론(Fuzzy set theory)에 기반하여 각 관측치가 여러 군집에 동시에 속할 수 있으며(data points can potentially belong to multiple clusters), 각 군집별로 속할 가능성(degrees of possibility, probability)을 제시해준다. 퍼지 군집 알고리즘으로 가장 널리 사용되는 것으로 FCM(Fuzzy C-Means) 클러스터링 알고리즘이다. FCM 알고리즘은 1973년 J.C.Dunn이 개발하였고, 1981년 J.C.Bezdek 이 발전시켰다.
FCM(Fuzzy C-Means) 클러스터링 방법은 하나의 클러스터에 속해져 있는 각각의 데이터 점을 소속도에 의해서 클러스터에 대한 데이터의 소속도를 일일이 열거한 데이터 분류 방법이다. FCM 방법의 특징은 모호한 것들을 명확히 표현하는 클러스터링 기법이고 한 데이터는 하나의 클러스터에 속할 수 있다. FCM 알고리즘은 다음과 같다.

[단계 1] 클러스터의 개수 c($2 \le c < n$)을 정하고 지수의 가중(exponential weight) m($1<m<\infty$)을 선택한다. 초기 소속함수 $U^{(0)}$를 초기화한다. 알고리즘 반복 횟수를 r(r=0, 1, 2, …)로 표시한다.

[단계 2] 식 (7.6)을 이용하여 퍼지 클러스터 중심{v_i | i=1, 2, …, c}을 계산한다.

$$ij = \frac{\sum_{k=1}^{n} (\mu_{jk})^m x_{kj}}{\sum_{k=1}^{n} (\mu_{ik})^m} \tag{7.6}$$

[단계 3] 다음과 같이 새로운 소속 함수 $U^{(r+1)}$을 계산한다.

$$u_{ik}^{(r+1)} = \frac{1}{\sum_{j=1}^{c} (\frac{d_{ik}^r}{d_{jk}^r})^{2/m-1}} \text{ for } I_k = \varnothing$$

또는

$$u_{ik}^{(r+1)} = 0 \text{ for all classes i, 여기서 } i \in \tilde{I}_k$$

여기서, I_k = {i|$2 \le c < n$; $d_{ik}^{(r)}$ = 0}이고 $\tilde{I}_k$ = {1, 2, …, c} - I_k이다. 그리고
$\sum_{i \in I_k} u_{ik}^{(r+1)} = 1$

[단계 4] 다음 식을 계산해서 만일 $\Delta > \epsilon$이면 r=r+1로 정하고 [단계 2]로 가서 다시 알고리즘을 반복 수행하고 그렇지 않고 $\Delta \le \epsilon$이면 알고리즘을 종료한다. 여기서, ϵ는 임계값.
$\triangle = \| U^{(r+1)} - U^{(r)} \| = \max_{i,k} | u_{ik}^{(r+1)} - u_{ik}^{(r)} |$

7.2.1 FCM 클러스터링의 수치적 예제

$x = \begin{bmatrix} 1 & 3 \\ 1.5 & 3.2 \\ 1.3 & 2.8 \\ 3 & 1 \end{bmatrix}$ 즉, x_1={1,3}, x_2={1.5,3.2}, x_3={1.3,2.8}, x_4={3,1}이라 할 때,

[단계 1] 클러스터 c=2, 지수의 가중(exponential weigh) m=2, 임계값 ϵ=0.01로 가정한 후, 초기 소속함수를 다음과 같이 정의한다.

$$U^{(0)} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

[단계 2] 각 클러스터에 대한 중심 벡터를 계산한다. 예제에서는 각 데이터 점에 대한 두 개의 좌표를 이루어져 있으므로 v_i={v_{i1}, v_{i2}}.

여기서, 첫 번째 클러스터에 대해서는(c=1), v_1={v_{11}, v_{12}}

두 번째 클러스터에 대해서는(c=2), v_2={v_{21}, v_{22}}

식 (7.6)에 의해서 풀어보면, $v_{ij} = \dfrac{\sum_{k=1}^{n} (u_{ik})^2 x_{kj}}{\sum_{k=1}^{n} (u_{ik})^2}$ (m=2이므로)

첫 번째 클러스터에서, v_{11}은, 즉 i=1, j=1, k=1, 2, 3, 4

$$v_{11} = \frac{u_{11}^2 x_{11} + u_{12}^2 x_{21} + u_{13}^2 x_{31} + u_{14}^2 x_{41}}{u_{11}^2 + u_{12}^2 + u_{13}^2 + u_{14}^2} = \frac{(1)1 + (1)1.6 + (1)1.3 + (0)3}{1+1+1+0} = \frac{3.8}{3} = 1.26$$

첫 번째 클러스터에서, v_{12}는, 즉 i=1, j=2, k=1, 2, 3, 4

$$v_{12} = \frac{u_{11}^2 x_{12} + u_{12}^2 x_{22} + u_{13}^2 x_{32} + u_{14}^2 x_{42}}{u_{11}^2 + u_{12}^2 + u_{13}^2 + u_{14}^2} = \frac{(1)3 + (1)3.2 + (1)2.8 + (0)1}{1+1+1+0} = \frac{9}{3} = 3.0$$

따라서 v_1={v_{11}, v_{12}}={1.26, 3.0}.

두 번째 클러스터에서, v_{21}은, 즉 i=2, j=1, k=1, 2, 3, 4

$$v_{21} = \frac{u_{21}^2 x_{11} + u_{22}^2 x_{21} + u_{23}^2 x_{31} + u_{24}^2 x_{41}}{u_{21}^2 + u_{22}^2 + u_{23}^2 + u_{24}^2} = \frac{(0)1 + (0)1.5 + (0)1.3 + (1)3}{0+0+0+1} = \frac{3}{1} = 3$$

두 번째 클러스터에서, v_{22}는, 즉 i=2, j=2, k=1, 2, 3, 4

$$v_{22} = \frac{u_{21}^2 x_{12} + u_{22}^2 x_{22} + u_{23}^2 x_{32} + u_{24}^2 x_{42}}{u_{21}^2 + u_{22}^2 + u_{23}^2 + u_{24}^2} = \frac{(0)3 + (0)3.2 + (0)2.8 + (1)1}{0+0+0+1} = \frac{1}{1} = 1$$

따라서 v_2={v_{21}, v_{22}}={3.0, 1.0}.

⊙ [단계 3] $u_{ik} = \dfrac{1}{\sum_{j=1}^{c}(\frac{d_{ik}}{d_{jk}})^2}$에 의해서 각 데이터들과 클러스터 중심과의 거리를 구한 후, 새로운 소속 행렬을 구성한다.

여기서, $d_{ik} = d(x_k - v_i) = [\sum_{j=1}^{1}(x_{kj} - v_{ij})^2]^{1/2}$

첫 번째 클러스터 중심과의 거리 계산	두 번째 클러스터 중심과의 거리 계산
$d_{1k} = [(x_{k1} - v_{11})^2 + (x_{k2} - v_{12})^2]^{1/2}$	$d_{2k} = [(x_{k1} - v_{21})^2 + (x_{k2} - v_{22})^2]^{1/2}$
$d_{11} = \sqrt{(1-1.26)^2 + (3-3)^2} = 0.26$ $d_{12} = \sqrt{(1.5-1.26)^2 + (3.2-3)^2} = 0.31$ $d_{13} = \sqrt{(1.3-1.26)^2 + (2.8-3)^2} = 0.20$ $d_{14} = \sqrt{(3-1.26)^2 + (1-3)^2} = 2.65$	$d_{21} = \sqrt{(1-3)^2 + (3-1)^2} = 2.82$ $d_{22} = \sqrt{(1.5-3)^2 + (3.2-1)^2} = 2.66$ $d_{23} = \sqrt{(1.3-3)^2 + (2.8-1)^2} = 2.47$ $d_{24} = \sqrt{(3-3)^2 + (1-1)^2} = 0$

중심과의 거리 계산을 바탕으로 $u_{ik}^{(r+1)} = \dfrac{1}{\sum_{j=1}^{c}(\frac{d_{ik}^{r}}{d_{jk}^{r}})^2}$ for $I_k = \varnothing$를 이용하여 새로운 소속 행렬 U를 생성한다.

$$u_{11} = [\sum_{j=1}^{c}(\frac{d_{11}}{d_{j1}})^2]^{-1} = [(\frac{d_{11}}{d_{11}})^2 + (\frac{d_{11}}{d_{21}})^2]^{-1} = [1 + (\frac{0.26}{2.82})^2]^{-1} = 0.991$$

$$u_{12} = [\sum_{j=1}^{c}(\frac{d_{12}}{d_{j2}})^2]^{-1} = [(\frac{d_{12}}{d_{12}})^2 + (\frac{d_{12}}{d_{22}})^2]^{-1} = [1 + (\frac{0.31}{2.66})^2]^{-1} = 0.986$$

$$u_{13} = [\sum_{j=1}^{c}(\frac{d_{13}}{d_{j3}})^2]^{-1} = [(\frac{d_{13}}{d_{13}})^2 + (\frac{d_{13}}{d_{23}})^2]^{-1} = [1 + (\frac{0.20}{2.47})^2]^{-1} = 0.993$$

$$u_{14} = [\sum_{j=1}^{c}(\frac{d_{14}}{d_{j4}})^2]^{-1} = [(\frac{d_{14}}{d_{14}})^2 + (\frac{d_{14}}{d_{24}})^2]^{-1} = [1 + (\frac{2.65}{0})^2]^{-1} \approx 0 \quad I_4 = \varnothing$$

$$u_{21} = [\sum_{j=1}^{c}(\frac{d_{21}}{d_{j1}})^2]^{-1} = [(\frac{d_{21}}{d_{11}})^2 + (\frac{d_{21}}{d_{21}})^2]^{-1} = [(\frac{2.82}{0.26})^2 + 1]^{-1} = 0.008$$

$$u_{22} = [\sum_{j=1}^{c}(\frac{d_{22}}{d_{j2}})^2]^{-1} = [(\frac{d_{22}}{d_{12}})^2 + (\frac{d_{22}}{d_{22}})^2]^{-1} = [(\frac{2.66}{0.31})^2 + 1]^{-1} = 0.014$$

$$u_{23} = [\sum_{j=1}^{c}(\frac{d_{23}}{d_{j3}})^2]^{-1} = [(\frac{d_{23}}{d_{13}})^2 + (\frac{d_{23}}{d_{23}})^2]^{-1} = [(\frac{2.47}{0.20})^2 + 1]^{-1} = 0.007$$

$$u_{24} = [\sum_{j=1}^{c}(\frac{d_{24}}{d_{j4}})^2]^{-1} = [(\frac{d_{24}}{d_{14}})^2 + (\frac{d_{24}}{d_{24}})^2]^{-1} = [(\frac{0.0}{2.65})^2 + 1]^{-1} = 1$$

따라서 갱신된 퍼지 소속 함수는 다음과 같이 된다.

$$U^{(1)} = \begin{bmatrix} 0.991 & 0.986 & 0.993 & 0 \\ 0.009 & 0.014 & 0.007 & 1 \end{bmatrix}$$

[단계 4] 식 $\triangle = \| U^{(r+1)} - U^{(r)} \| = \max_{i,k} | u_{ik}^{(r+1)} - u_{ik}^{(r)} |$에서 $\max_{i,k}$값이 주어진 임계값보다 작으면 종료판정을 한다. $\max_{i,k} | u_{ik}^{(1)} - u_{ik}^{(0)} | = 0.0134 > 0.01$이므로 r=r+1로 놓고 [단계 2]로 넘어간다.

[단계 2]
첫 번째 클러스터에 대해서는(c=1). v_1={v_{11}, v_{12}}
두 번째 클러스터에 대해서는(c=1). v_2={v_{21}, v_{22}}

식 (7,6)에 의해서 풀어보면, $v_{ij} = \dfrac{\sum_{k=1}^{n} (u_{ik})^2 x_{kj}}{\sum_{k=1}^{n} (u_{ik})^2}$ (m=2이므로)

첫 번째 클러스터에서, v_{11}은, 즉 i=1, j=1, k=1, 2, 3, 4

$$v_{11} = \frac{u_{11}^2 x_{11} + u_{12}^2 x_{21} + u_{13}^2 x_{31} + u_{14}^2 x_{41}}{u_{11}^2 + u_{12}^2 + u_{13}^2 + u_{14}^2}$$
$$= \frac{(0.991)1 + (0.986)1.6 + (0.993)1.3 + (0)3}{0.991 + 0.986 + 0.993 + 0} = \frac{3.72}{2.97} = 1.25$$

첫 번째 클러스터에서, v_{12}은, 즉 i=1, j=2, k=1, 2, 3, 4

$$v_{12} = \frac{u_{11}^2 x_{12} + u_{12}^2 x_{22} + u_{13}^2 x_{32} + u_{14}^2 x_{42}}{u_{11}^2 + u_{12}^2 + u_{13}^2 + u_{14}^2}$$
$$= \frac{(0.991)3 + (0.986)3.2 + (0.993)2.8 + (0)1}{0.991 + 0.986 + 0.993 + 0} = \frac{8.82}{2.97} = 2.97$$

따라서 v_1={v_{11}, v_{12}}={1.25, 2.97}.

두 번째 클러스터에서, v_{21}은, 즉 i=2, j=1, k=1, 2, 3, 4

$$v_{21} = \frac{u_{21}^2 x_{11} + u_{22}^2 x_{21} + u_{23}^2 x_{31} + u_{24}^2 x_{41}}{u_{21}^2 + u_{22}^2 + u_{23}^2 + u_{24}^2}$$
$$= \frac{(0.009)1 + (0.014)1.5 + (0.007)1.3 + (1)3}{0.009 + 0.014 + 0.007 + 1} = \frac{3}{1.03} = 2.91$$

두 번째 클러스터에서, v_{22}은, 즉 i=2, j=2, k=1, 2, 3, 4

$$v_{22} = \frac{u_{21}^2 x_{12} + u_{22}^2 x_{22} + u_{23}^2 x_{32} + u_{24}^2 x_{42}}{u_{21}^2 + u_{22}^2 + u_{23}^2 + u_{24}^2}$$
$$= \frac{(0.009)3 + (0.014)3.2 + (0.007)2.8 + (1)1}{0.009 + 0.014 + 0.007 + 1} = \frac{1}{1.03} = 0.97$$

따라서 v_2={v_{21}, v_{22}}={2.91, 0.97}.

[단계 3] $u_{ik} = \frac{1}{\sum_{j=1}^{c}(\frac{d_{ik}}{d_{jk}})^2}$에 의해서 각 데이터들과 클러스터 중심과의 거리를 구한 후, 새로운 소속 행렬을 구성한다.

여기서, $d_{ik} = d(x_k - v_i) = [\sum_{j=1}^{1}(x_{kj} - v_{ij})^2]^{1/2}$

첫 번째 클러스터 중심과의 거리 계산	두 번째 클러스터 중심과의 거리 계산
$d_{1k} = [(x_{k1} - v_{11})^2 + (x_{k2} - v_{12})^2]^{1/2}$	$d_{2k} = [(x_{k1} - v_{21})^2 + (x_{k2} - v_{22})^2]^{1/2}$
$d_{11} = \sqrt{(1-1.25)^2 + (3-2.97)^2} = 0.25$	$d_{21} = \sqrt{(1-2.91)^2 + (3-0.97)^2} = 2.79$
$d_{12} = \sqrt{(1.5-1.25)^2 + (3.2-2.97)^2} = 0.34$	$d_{22} = \sqrt{(1.5-2.91)^2 + (3.2-0.97)^2} = 2.64$
$d_{13} = \sqrt{(1.3-1.25)^2 + (2.8-2.97)^2} = 0.18$	$d_{23} = \sqrt{(1.3-2.91)^2 + (2.8-0.97)^2} = 2.44$
$d_{14} = \sqrt{(3-1.25)^2 + (1-2.97)^2} = 2.63$	$d_{24} = \sqrt{(3-2.91)^2 + (1-0.97)^2} = 0.09$

중심과의 거리 계산을 바탕으로 $u_{ik}^{(r+1)} = \frac{1}{\sum_{j=1}^{c}(\frac{d_{ik}^r}{d_{jk}^r})^2}$ for $I_k = \varnothing$를 이용하여 새로운 소속 행렬 U를 생성한다.

$$u_{11} = [\sum_{j=1}^{c}(\frac{d_{11}}{d_{j1}})^2]^{-1} = [(\frac{d_{11}}{d_{11}})^2 + (\frac{d_{11}}{d_{21}})^2]^{-1} = [1 + (\frac{0.25}{2.79})^2]^{-1} = 0.992$$

$$u_{12} = [\sum_{j=1}^{c}(\frac{d_{12}}{d_{j2}})^2]^{-1} = [(\frac{d_{12}}{d_{12}})^2 + (\frac{d_{12}}{d_{22}})^2]^{-1} = [1 + (\frac{0.34}{2.64})^2]^{-1} = 0.984$$

$$u_{13} = [\sum_{j=1}^{c}(\frac{d_{13}}{d_{j3}})^2]^{-1} = [(\frac{d_{13}}{d_{13}})^2 + (\frac{d_{13}}{d_{23}})^2]^{-1} = [1 + (\frac{0.18}{2.44})^2]^{-1} = 0.995$$

$$u_{14} = [\sum_{j=1}^{c}(\frac{d_{14}}{d_{j4}})^2]^{-1} = [(\frac{d_{14}}{d_{14}})^2 + (\frac{d_{14}}{d_{24}})^2]^{-1} = [1 + (\frac{2.63}{0.09})^2]^{-1} \approx 0 \quad I_4 = \varnothing$$

$$u_{21} = [\sum_{j=1}^{c}(\frac{d_{21}}{d_{j1}})^2]^{-1} = [(\frac{d_{21}}{d_{11}})^2 + (\frac{d_{21}}{d_{21}})^2]^{-1} = [(\frac{2.79}{0.25})^2 + 1]^{-1} = 0.008$$

$$u_{22} = [\sum_{j=1}^{c}(\frac{d_{22}}{d_{j2}})^2]^{-1} = [(\frac{d_{22}}{d_{12}})^2 + (\frac{d_{22}}{d_{22}})^2]^{-1} = [(\frac{2.64}{0.34})^2 + 1]^{-1} = 0.016$$

$$u_{23} = [\sum_{j=1}^{c}(\frac{d_{23}}{d_{j3}})^2]^{-1} = [(\frac{d_{23}}{d_{13}})^2 + (\frac{d_{23}}{d_{23}})^2]^{-1} = [(\frac{2.44}{0.18})^2 + 1]^{-1} = 0.005$$

$$u_{24} = [\sum_{j=1}^{c}(\frac{d_{24}}{d_{j4}})^2]^{-1} = [(\frac{d_{24}}{d_{14}})^2 + (\frac{d_{24}}{d_{24}})^2]^{-1} = [(\frac{0.09}{2.63})^2 + 1]^{-1} = 1$$

따라서 갱신된 퍼지 소속 함수는 다음과 같이 된다.

$$U^{(2)} = \begin{bmatrix} 0.992 & 0.984 & 0.995 & 0 \\ 0.008 & 0.016 & 0.005 & 1 \end{bmatrix}$$

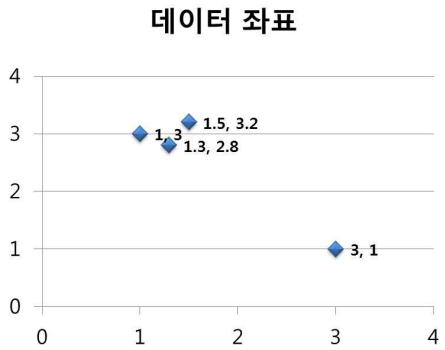
[단계 4]

식 $\triangle = \| U^{(r+1)} - U^{(r)} \| = \max_{i,k} | u_{ik}^{(r+1)} - u_{ik}^{(r)} |$에서 $\max_{i,k}$값이 주어진 임계값보다 작으면 종료판정을 한다. $\max_{i,k} | u_{ik}^{(2)} - u_{ik}^{(1)} | = 0.002 < 0.01$이므로 알고리즘을 종료한다.

따라서 최종 소속 행렬은 $U^{(2)} = \begin{bmatrix} 0.992 & 0.984 & 0.995 & 0 \\ 0.008 & 0.016 & 0.005 & 1 \end{bmatrix}$

중심 값은 v_1={v_{11}, v_{12}}={1.25, 2.97}, v_2={v_{21}, v_{22}}={2.91, 0.97}.

7.2.2 C# FCM 소스 프로그램



fcm.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace fcm
{

    class fcm
    {
        const int DATA = 4;
        const int CLUSTER = 2;
        const int COORD = 2;
        const int w_para = 2;
        const double Error = 0.01;  //임계값
```

```
public static double[,] U = new double[CLUSTER, DATA];
public static double[,] U_old = new double[CLUSTER, DATA];
public static double[,] v = new double[CLUSTER, COORD];
public static double[,] d = new double[CLUSTER, DATA];
public static double[,] x = new double[DATA, COORD];

int i, j, k, iter;
double[,] Rand_U = new double[DATA, CLUSTER];
double[] Init_Sum = new double[DATA];
double[,] Temp_U = new double[DATA, CLUSTER];
double num, den, t_num, t_den;
double temp1_dist, temp2_dist;
double[,] temp_e = new double[CLUSTER, DATA];
double max_error;
double sum, NewValue;
int count, sing;
int cn = 0;

Random rr = new Random();
public fcm()      //생성자
{
    double[] u = new double[8];
    Console.WriteLine("FCM 프로그램입니다. X,Y를 순서대로 입력하시오.");
    for (i = 0; i < 8; i++)
    {
        u[i] = double.Parse(Console.ReadLine());
    }

    for (i = 0; i < 4; i++)
        for (j = 0; j < 2; j++)
        {
            x[i, j] = u[cn];
            cn++;
            Console.WriteLine("["+i+"]" + "[" + j + "] = " + x[i, j]);
        }

    u_matrix();


    do
```

```
        {
            cal_vector();
            cal_center();
            cal_update();
            cal_error();
        } while (count != 0);
    }

    // Initialize Make U-matrix
    public void u_matrix()
    {
        for (i = 0; i < DATA; i++)
        {
            U[0, i] = 1;
            U[0, 3] = 0;
        }
        for (i = 0; i < DATA; i++)
        {
            U[1, i] = 0;
            U[1, 3] = 1;
        }
        Console.WriteLine("소속 초기행렬");
        for (i = 0; i < CLUSTER; i++)
        {
            for (j = 0; j < DATA; j++)
            {
                Console.Write(U[i, j]);
            }
            Console.Write("\n");
        }
        Console.Write("\n");
        iter = 0;
    }




    //Calculate the center v
```

```
public void cal_vector()
{
    for (i = 0; i < CLUSTER; i++)
    {
        for (j = 0; j < COORD; j++)
        {
            num = 0;
            den = 0;
            for (k = 0; k < DATA; k++)
            {
                t_num = 0;
                t_den = 0;
                t_num = Math.Pow(U[i, k], w_para) * x[k, j];
                t_den = Math.Pow(U[i, k], w_para);
                Console.Write("\n t_num =" + t_num + " t_den =" + t_den);
                num += t_num;
                den += t_den;
            }
            v[i, j] = num / den;
            Console.Write("\n");
        }
        Console.Write("\n");
    }
    Console.Write("\n");

    for (i = 0; i < CLUSTER; i++)
    {
        for (j = 0; j < COORD; j++)
        {
            Console.Write(v[i, j] + "   ");
        }
        Console.Write("\n");
    }
    Console.Write("\n");

}



//Calculate distance between data and cluster center
```

```
public void cal_center()
{
    for (i = 0; i < CLUSTER; i++)
    {
        for (j = 0; j < DATA; j++)
        {
            temp2_dist = 0;
            for (k = 0; k < COORD; k++)
            {
                temp1_dist = Math.Pow((x[j, k] - v[i, k]), 2);
                temp2_dist += temp1_dist;
            }
            d[i, j] = Math.Sqrt(temp2_dist);
        }
    }
    for (i = 0; i < CLUSTER; i++)
    {
        for (j = 0; j < DATA; j++)
        {
            Console.Write("d=   " + d[i, j] + "\n");
        }
    }
    Console.Write("\n");

}

public void cal_update()
{
    for (k = 0; k < DATA; k++)
    {
        for (i = 0; i < CLUSTER; i++)
        {
            if (d[i, k] != 0)
            {
                for (j = 0, sum = 0; j < COORD; j++)
                {
                    if (i == j) sum += 1.0;
                    else if (d[j, k] == 0)
                    {
                        U[i, k] = 0.0;
```

```
                        break
                    }
                    else sum += Math.Pow(d[i, k] / d[j, k], 2 / (w_para - 1));
                }
                NewValue = 1.0 / sum;
                U[i, k] = NewValue;
               Console.Write("U" + "[" + i + "]" + "[" + k + "] = " + NewValue);
                Console.Write("\n");
            }
            else
            {
                for (j = 0, sing = 1; j < i; j++) U[j, k] = 0.0;
                  for (j = i + 1, sing = 1; j < COORD; j++)
                    if (d[j, k] == 0) sing++;
                                U[i, k] = 1.0 / sing;
                // Console.Write(" U[" + i + "][" + k + "] = " + U[i, k]);
                for (j = i + 1; j < CLUSTER; j++)
                {
                    if (d[j, k] == 0) U[i, k] = 1.0 / sing;
                    else U[i, k] = 0.0;
                }
                break;
            }
        }
        break;
    }
    Console.Write("\n");
}

public void cal_error()
{
    max_error = 0.0;
    for (i = 0; i < CLUSTER; i++)
    {
        for (k = 0; k < DATA; k++)
        {
            temp_e[i, k] = Math.Abs(U[i, k] - U_old[i, k]);
            max_error = Math.Max(max_error, temp_e[i, k]);
        }
    }
    for (i = 0; i < CLUSTER; i++)
```

```
            {
                for (k = 0; k < DATA; k++)
                {
                    U_old[i, k] = U[i, k];
                }
            }
            Console.Write("\n error = " + max_error);
            count = 0;
            if (max_error > Error) count = count + 1;
            iter++;
        }

        public void print()
        {
            Console.Write("\n\n*****final CLuster Center *****\n");
            for (i = 0; i < CLUSTER; i++)
            {
                for (j = 0; j < COORD; j++)
                {
                    Console.Write("   v[" + i + "][" + j + "] = " + v[i, j]);
                }
                Console.Write("\n");
            }
            Console.Write("\n");
            Console.Write("Iteration = " + iter + "    error =" + max_error);

        }

    }
    class Program
    {
        static void Main(string[] args)
        {

            fcm gy = new fcm();
            gy.print();
        }
    }
}
```