FUZZY C-MEANS CLUSTERING

Fuzzy c-means (FCM) algoritması, bulanık bölünmeli kümeleme tekniklerinden en iyi bilinen ve yaygın kullanılan yöntemdir. Fuzzy c-means metodu nesnelerin iki veya daha fazla kümeye ait olabilmesine izin verir. Bulanık mantık prensibi gereği her veri, kümelerin her birine [0,1] arasında değişen birer üyelik değeri ile aittir. Bir verinin tüm sınıflara olan üyelik değerleri toplamı "1" olmalıdır. Nesne hangi küme merkezine yakın ise o kümeye ait olma üyeliği diğer kümelere ait olma üyeliğinden daha büyük olacaktır.

Fuzzy c-means algoritmasının en önemli özelliği olan üyelik matrisinin kümeleme üzerinde olumlu etkileri vardır. Bu matris belirsiz durumların tanımlanmasını kolaylaştırır. Ayrıca üyelik dereceleri düşük olduğundan sıra dışı verilerin etkisi azdır. Esnek bir yapıya sahiptir. Örtüşen kümeleri bulma kabiliyeti diğer bölünmeli algoritmalara göre daha fazladır.

Yukarıda bahsedilen avantajların yanında fuzzy c-means algoritmasının bazı dezavantajları da vardır. Üyelik fonksiyonu işlemsel karmaşıklığı arttırdığı için zaman açısından maliyetli bir bölünmeli kümeleme algoritmasıdır.

Fuzzy c-means algoritmasında amaç fonksiyonu temelli bir metottur. Algoritma, en küçük kareler yönteminin genellemesi olan aşağıdaki amaç fonksiyonunu öteleyerek minimize etmek için çalışır.

$$Jm = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^{m} \left\| x_i - c_j \right\|^2 \quad ,1 \leq m < \infty \quad (2)$$

U üyelik matrisi rastgele atanarak algoritma başlatılır. İkinci adımda ise merkez vektörleri hesaplanır. Merkezler (3)'deki formüle göre hesaplanır.

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} x_i}{\sum_{i=1}^{N} u_{ij}^{m}} \quad (3)$$

Hesaplanan küme merkezlerine göre U matrisi aşağıdaki formül kullanılarak yeniden hesaplanır. Eski U matrisi ile yeni U matrisi karşılaştırılır ve fark ε'dan küçük olana kadar işlemler devam eder.

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\left\| x_i - c_i \right\|}{\left\| x_i - c_k \right\|} \right)^{2/(m-1)}} \quad (4)$$

Kümeleme işlemi sonucunda bulanık değerler içeren U üyelik matrisi kümelemenin sonucunu yansıtır. İstenirse, berraklaştırma yapılarak bu değerler yuvarlanıp 0 ve 1'lere dönüştürülebilir.

Şimdi bu yöntemi kullanarak geliştirdiğim veri madenciliği yöntemimin kodunu paylaşıyorum. Uygulamada kullanılan dataset eğitim amaçlıdır.

loadTest.m dosyamın içeriği

```
function [dataA,dataB,dataC,dataD,dataE,dataTarget] = loadTestset;
% Load data
filename = 'Testset.xls';
dataA = xlsread(filename,'B2:B2001');
dataB = xlsread(filename,'C2:C2001');
dataC = xlsread(filename,'D2:D2001');
dataD = xlsread(filename,'E2:E2001');
dataE = xlsread(filename,'F2:F2001');
dataTarget = xlsread(filename,'G2:G2001');
```

loadTrainset.m dosyasının içeriği

```
function [dataA,dataB,dataC,dataD,dataE,dataTarget] = loadTrainset;
% Load data
filename = 'Trainset.xls';
dataA = xlsread(filename,'B2:B6001');
dataB = xlsread(filename,'C2:C6001');
dataC = xlsread(filename,'D2:D6001');
dataD = xlsread(filename,'E2:E6001');
dataE = xlsread(filename,'F2:F6001');
dataTarget = xlsread(filename,'G2:G6001');
```

fcm_SC.m dosyasının içeriği

```
close all;
clear all;
clc;

% Load data train

[dataA,dataB,dataC,dataD,dataE,dataTarget] = loadTrainset;
M = [dataA,dataB,dataC,dataD,dataE];

% Find the centroid from data testing
[centers,U] = fcm(M,8);
maxU = max(U);

% Spare into 8 clusters
index1 = find(U(1,:) == maxU);
index2 = find(U(2,:) == maxU);
index3 = find(U(3,:) == maxU);
index4 = find(U(4,:) == maxU);
index5 = find(U(5,:) == maxU);
index6 = find(U(6,:) == maxU);
index7 = find(U(7,:) == maxU);
index8 = find(U(8,:) == maxU);

% Show the clusters
hold on
plot(M(index1,1),M(index1,2),'ob')
plot(M(index2,1),M(index2,2),'or')
plot(M(index3,1),M(index3,2),'og')
plot(M(index4,1),M(index4,2),'oy')
plot(M(index5,1),M(index5,2),'om')
```

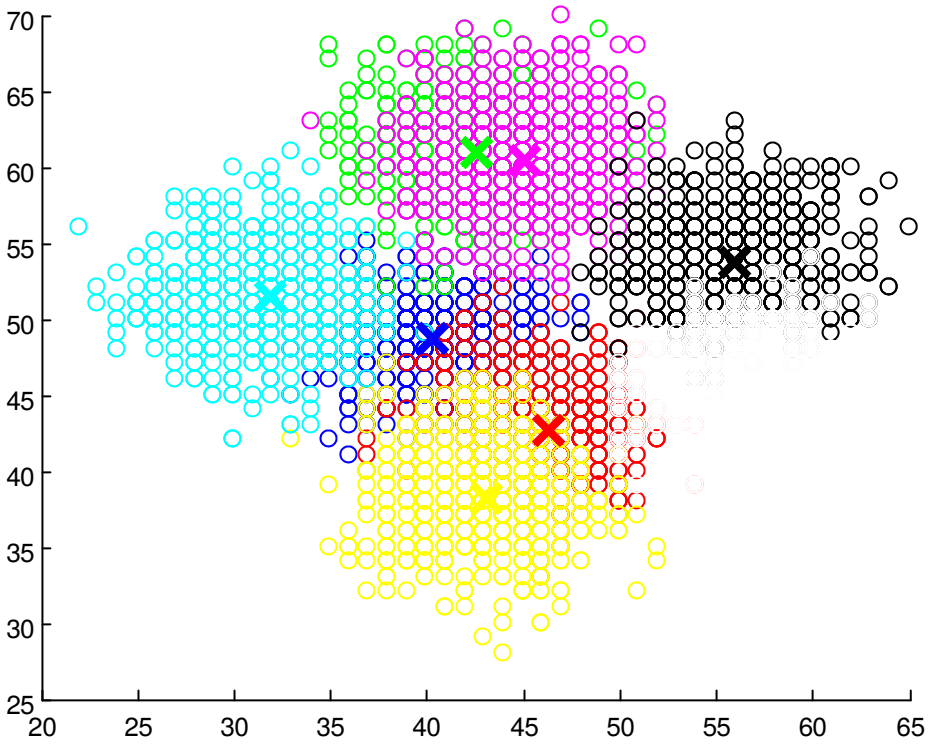
```
plot(M(index6,1),M(index6,2),'oc')
plot(M(index7,1),M(index7,2),'ok')
plot(M(index8,1),M(index8,2),'ow')
plot(centers(1,1),centers(1,2),'xb','MarkerSize',15,'LineWidth',3)
plot(centers(2,1),centers(2,2),'xr','MarkerSize',15,'LineWidth',3)
plot(centers(3,1),centers(3,2),'xg','MarkerSize',15,'LineWidth',3)
plot(centers(4,1),centers(4,2),'xy','MarkerSize',15,'LineWidth',3)
plot(centers(5,1),centers(5,2),'xm','MarkerSize',15,'LineWidth',3)
plot(centers(6,1),centers(6,2),'xc','MarkerSize',15,'LineWidth',3)
plot(centers(7,1),centers(7,2),'xk','MarkerSize',15,'LineWidth',3)
plot(centers(8,1),centers(8,2),'xw','MarkerSize',15,'LineWidth',3)
hold off

% Count majority of class from index1
nul1 = 0;
one1 = 0;
for i = 1:size(index1,1)
   if dataTarget(index1(i)) == 0
      nul1 = nul1 + 1;
   else
      one1 = one1 + 1;
   end
end

% Count majority of class from index2
nul2 = 0;
one2 = 0;
for i = 1:size(index2,1)
   if dataTarget(index2(i)) == 0
      nul2 = nul2 + 1;
   else
      one2 = one2 + 1;
   end
end

% Count majority of class from index3
nul3 = 0;
one3 = 0;
for i = 1:size(index3,1)
   if dataTarget(index3(i)) == 0
      nul3 = nul3 + 1;
   else
      one3 = one3 + 1;
   end
end

% Count majority of class from index4
nul4 = 0;
one4 = 0;
for i = 1:size(index4,1)
   if dataTarget(index4(i)) == 0
      nul4 = nul4 + 1;
   else
      one4 = one4 + 1;
   end
end
```

```
% Count majority of class from index5
nul5 = 0;
one5 = 0;
for i = 1:size(index5,1)
   if dataTarget(index5(i)) == 0
      nul5 = nul5 + 1;
   else
      one5 = one5 + 1;
   end
end

% Count majority of class from index6
nul6 = 0;
one6 = 0;
for i = 1:size(index6,1)
   if dataTarget(index6(i)) == 0
      nul6 = nul6 + 1;
   else
      one6 = one6 + 1;
   end
end

% Count majority of class from index7
nul7 = 0;
one7 = 0;
for i = 1:size(index7,1)
   if dataTarget(index7(i)) == 0
      nul7 = nul7 + 1;
   else
      one7 = one7 + 1;
   end
end

% Count majority of class from index8
nul8 = 0;
one8 = 0;
for i = 1:size(index8,1)
   if dataTarget(index8(i)) == 0
      nul8 = nul8 + 1;
   else
      one8 = one8 + 1;
   end
end

% Labelling each cluster
label = zeros(8,1,'uint32');
if nul1>=one1
   label(1) = 0;
else label(1) = 1;
end
if nul2>=one2
   label(2) = 0;
else label(2) = 1;
end
if nul3>=one3
   label(3) = 0;
else label(3) = 1;
```

```
end
if nul4>=one4
   label(4) = 0;
else label(4) = 1;
end
if nul5>=one5
   label(5) = 0;
else label(5) = 1;
end
if nul6>=one6
   label(6) = 0;
else label(6) = 1;
end
if nul7>=one7
   label(7) = 0;
else label(7) = 1;
end
if nul8>=one8
   label(8) = 0;
else label(8) = 1;
end

% Find the centroid from data testing part 2
[dataA,dataB,dataC,dataD,dataE,dataTarget] = loadTestset;
M = [dataA,dataB,dataC,dataD,dataE];
[centers2,U2] = fcm(M,8);
maxU2 = max(U2);

% Spare into 8 clusters
index1_2 = find(U2(1,:) == maxU2);
index2_2 = find(U2(2,:) == maxU2);
index3_2 = find(U2(3,:) == maxU2);
index4_2 = find(U2(4,:) == maxU2);
index5_2 = find(U2(5,:) == maxU2);
index6_2 = find(U2(6,:) == maxU2);
index7_2 = find(U2(7,:) == maxU2);
index8_2 = find(U2(8,:) == maxU2);

sumin = zeros(8,2,'uint32');
for i = 1:8
   for j = 1:5
      sumin(i,1) = sumin(i,1) + centers(i,j); sumin(i,2) = sumin(i,2) + centers2(i,j);
   end
end

% Find the nearest centroid
kelas = zeros(8,2,'uint32');
x = distfcm(centers,M(i,:));
for i = 1:8
   for j = 1:8
      if (j == 1)
         kelas(i,1) = abs(double(sumin(j,1))-double(sumin(i,2)));
         x = j;
      else
         if kelas(i,1)>abs(double(sumin(j,1))-double(sumin(i,2)))
            kelas(i,1) = abs(double(sumin(j,1))-double(sumin(i,2)));
            x = j;
```

```
            end
        end
    end
    kelas(i,2) = label(x);
end

tebakan = zeros(2000,1,'uint32');
for i = 1:size(index1_2,1)
    tebakan(index1_2(i)) = kelas(1,2);
end
for i = 1:size(index2_2,1)
    tebakan(index2_2(i)) = kelas(2,2);
end
for i = 1:size(index3_2,1)
    tebakan(index3_2(i)) = kelas(3,2);
end
for i = 1:size(index4_2,1)
    tebakan(index4_2(i)) = kelas(4,2);
end
for i = 1:size(index5_2,1)
    tebakan(index5_2(i)) = kelas(5,2);
end
for i = 1:size(index6_2,1)
    tebakan(index6_2(i)) = kelas(6,2);
end
for i = 1:size(index7_2,1)
    tebakan(index7_2(i)) = kelas(7,2);
end
for i = 1:size(index8_2,1)
    tebakan(index8_2(i)) = kelas(8,2);
end

indexX = zeros(1,6000,'uint32');

% for i = 1:2000
%     x = distfcm(centers,M(i,:));
% %     if (x(2)>x(1))
% %         indexX(i) = 1;
% %     end
% end

ansTrue = 0;
for i = 1:2000
    if (tebakan(i) == dataTarget(i))
        ansTrue = ansTrue + 1;
    end
end
accuracy = ansTrue/20
```

Çıktısı aşağıdaki gibidir….


70
65
60
55
50
45
40
35
30
25
20
25
30
35
40
45
50
55
60
65