به نام خدا

الگوریتم های بررسی شده در مقاله به منظور کلاسترینگ به شرح زیر می باشد که توضیحات مربوط به شبیه سازی هر کدام جداگانه آورده می شود.

4.2. Fuzzy clustering

یکی از معروف ترین روش ها برای کلاسترینگ فازی Fuzzy cmeans می باشد. که فرمول های به روز رسانی میانگین و مرکز دسته ها و تابع هزینه به صورت زیر است:

4.2.1. Fuzzy C-means

$$J_m = \sum_{i=1}^{c}\sum_{j=1}^{n}\mu_{ij}^{m}||x_i - c_i||^2 \qquad (2)$$

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C}(||x_i - c_i||/x_j - c_k)^{2/(m-1)}} \qquad (3)$$

The cluster center c_i can be obtained from:

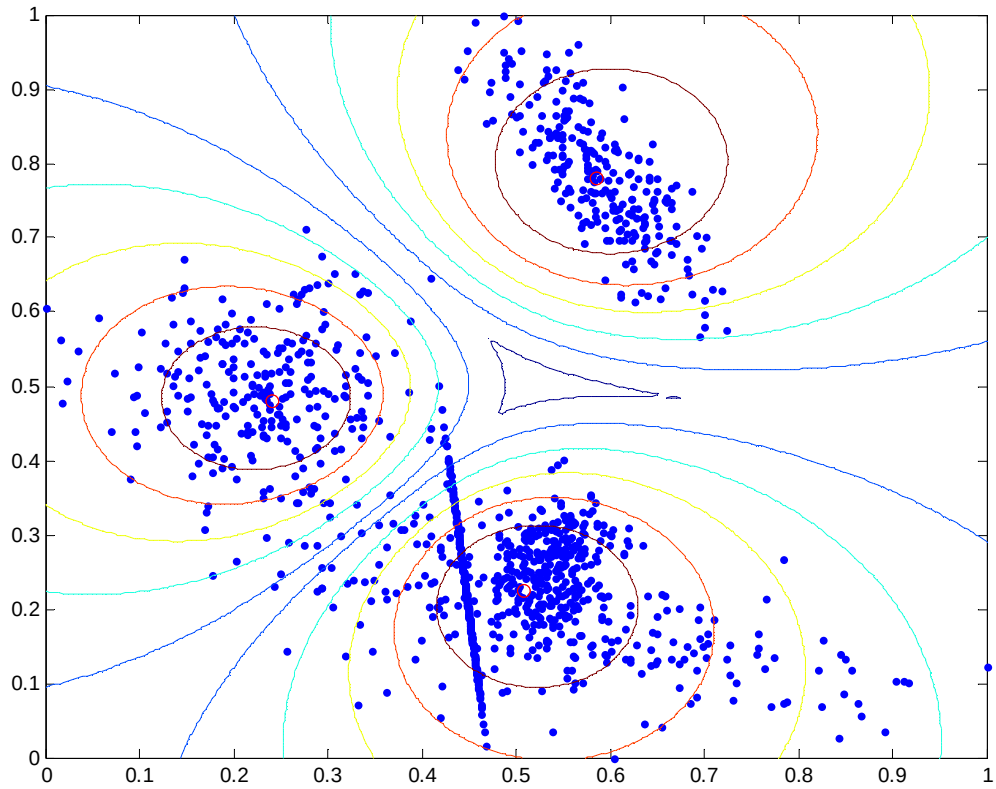
$$c_i = \frac{\sum_{j=1}^{N}\mu_{ij}^{m}x_j}{\sum_{j=1}^{N}\mu_{ij}^{m}} \qquad (4)$$

مهمترین قسمت برنامه در تابع FCMclust است

```
function result = FCMclust(data,param)


%data normalization
                                                    داده های ورودی
X=data.X;
                                  در هر مساله کلاسترینگ باید تعداد کلاس ها را مشخص نمود.
f0=param.c;


[N,n] = size(X);
[Nf0,nf0] = size(f0);
X1 = ones(N,1);

% Initialize fuzzy partition matrix
rand('state',0)
if max(Nf0,nf0) == 1,       % only number of cluster given
  c = f0;
```

```
result.data.d=distout;
result.cluster.v=v;
result.iter = iter;
result.cost = J;
```

با اجرای فایل FCMcall در پوشه main به نتیجه زیر می رسیم



الگوریتم دوم مطرح شده به صورت زیر است:

4.2.2. *Gustafson–Kessel algorithm*

The algorithm of Gustafson–Kessel fuzzy clustering (GK-clust)

```
  iter = iter + 1;
  % Calculate centers
  f = f0;
  fm = f.^m;
  sumf = sum(fm);

  v = (fm'*X)./(sumf'*ones(1,n));

  for j = 1 : c,
    xv = X - X1*v(j,:);
    % Calculate covariance matrix
```

Step 2: Compute the cluster covariance matrices

```
    A = ones(n,1)*fm(:,j)'.*xv'*xv/sumf(j);
    Pi(:,:,j)=1/N*sum(fm(:,j));
    d(:,j) =
1/(det(pinv(A))^(1/2))*1/Pi(:,:,j)*exp(1/2*sum((xv*pinv(A).*xv),2));
    %d(:,j) = 1/(det(A)^(1/2))*Pi(:,:,j)*exp(-1/2*sum((xv*pinv(A).*xv)'));
end
  distout=sqrt(d);
  % Update f0
  if m>1                                %Gath-Geva clustering
      d = (d+1e-10).^(-1/(m-1));
  else                                  %Fuzzy Maximum Likelihood Estimates
clustering
      d = (d+1e-10).^(-1);
  end
      f0 = (d ./ (sum(d,2)*ones(1,c)));
      J(iter) = sum(sum(f0.*d));           %calculate objective function
```

$$J(Z;U,V,\{A_i\}) = \sum_{i=1}^{K}\sum_{k=1}^{N}(\mu_{ik})^m D^2_{ikA_i} \qquad (5)$$

```
end


fm = f;sumf = sum(fm);

P = zeros(n,n,c);             % covariance matrix
V = zeros(c,n);                % eigenvectors
D = V;                          % eigenvalues

% calculate P,V,D
for j = 1 : c,
    xv = X - ones(N,1)*v(j,:);
% Calculate covariance matrix

    A = ones(n,1)*fm(:,j)'.*xv'*xv/sumf(j);
% Calculate eigen values and eigen vectors
    [ev,ed] = eig(A); ed = diag(ed)';
    ev = ev(:,ed == min(ed));
% Put cluster info in one matrix
    P(:,:,j) = A;
    V(j,:) = ev';
    D(j,:) = ed;
end;
```

Repeat for $l = 1, 2, \ldots$

Step 1: Compute cluster prototypes (means):

$$\mathbf{v}_i^{(l)} = \frac{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^m \mathbf{z}_k}{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^m}, \quad 1 \le i \le K\,.$$

Step 2: Compute the cluster covariance matrices:

$$\mathbf{F}_i = \frac{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^m (\mathbf{z}_k - \mathbf{v}_i^{(l)})(\mathbf{z}_k - \mathbf{v}_i^{(l)})^T}{\sum_{k=1}^{N} (\mu_{ik}^{(l-1)})^m},$$
$$1 \le i \le K\,.$$

Step 3: Compute the distances:

$$D^2_{ik\mathbf{A}_i} = (\mathbf{z}_k - \mathbf{v}_i^{(l)})^T \left[\rho_i \det(\mathbf{F}_i)^{1/n} \mathbf{F}_i^{-1}\right] (\mathbf{z}_k - \mathbf{v}_i^{(l)})$$
$$1 \le i \le K, \quad 1 \le k \le N\,.$$

Step 4: Update the partition matrix:

for $1 \le k \le N$

if $D_{ik\mathbf{A}_i} > 0$ for $1 \le i \le K$,

$$\mu_{ik}^{(l)} = \frac{1}{\sum_{j=1}^{K} (D_{ik\mathbf{A}_i} / D_{jk\mathbf{A}_j})^{2/(m-1)}},$$

otherwise

$$\mu_{ik}^{(l)} = 0 \text{ if } D_{ik\mathbf{A}_i} > 0, \text{ and } \mu_{ik}^{(l)} \in [0, 1]$$
$$\text{with} \sum_{i=1}^{K} \mu_{ik}^{(l)} = 1 \text{ otherwise}\,.$$

until $\|\mathbf{U}^{(l)} - \mathbf{U}^{(l-1)}\| < \epsilon$.

Step 3: Compute the distances:

$$D^2_{ik\mathbf{A}_i} = (\mathbf{z}_k - \mathbf{v}_i^{(l)})^T \left[\rho_i \det(\mathbf{F}_i)^{1/n}\mathbf{F}_i^{-1}\right](\mathbf{z}_k - \mathbf{v}_i^{(l)}),$$
$$1 \le i \le K, \quad 1 \le k \le N.$$

```
    M = (1/det(pinv(A))/rho(j))^(1/n)*pinv(A);
    %M(:,:,j) = (det(A)/rho(j)).^(1/n)*pinv(A);
    d(:,j) = sum((xv*M.*xv),2);
  end

  distout=sqrt(d);

  J(iter) = sum(sum(f0.*d));           %calculate objective function
  % Update f0
  d = (d+1e-10).^(-1/(m-1));
```

$$\mu_{ik}^{(l)} = \frac{1}{\sum_{j=1}^{K}(D_{ik\mathbf{A}_i}/D_{jk\mathbf{A}_j})^{2/(m-1)}},$$

```
  f0 = (d ./ (sum(d,2)*ones(1,c)));


end             %end of iteration period

fm = f.^m; sumf = sum(fm);

P = zeros(n,n,c);             % covariance matrix
M = P;                          % norm-inducing matrix
V = zeros(c,n);                % eigenvectors
D = V;                          % eigenvalues

% calculate P,V,D,M
for j = 1 : c,
    xv = X - ones(N,1)*v(j,:);
    % Calculate covariance matrix
    A = ones(n,1)*fm(:,j)'.*xv'*xv/sumf(j);
    % Calculate eigen values and eigen vectors
    [ev,ed] = eig(A); ed = diag(ed)';
    ev = ev(:,ed == min(ed));
    % Put cluster info in one matrix
    P(:,:,j) = A;
    M(:,:,j) = (det(A)/rho(j)).^(1/n)*pinv(A);
    V(j,:) = ev';
    D(j,:) = ed;
end
```