Implementation of Possibilistic Fuzzy C-Means Clustering Algorithm in Matlab

Neelam Kumari, Bhawna Sharma, Dr. Deepti Gaur
Dept. of Computer Science & IT, ITM University, Gurgaon, INDIA.
singh.neelam693@gmail.com
bhawnash.6@gmail.com
deepi_k_Gaur@yahoo.com

Abstract—Clustering means classifying the given observation data sets into subgroups or clusters. It is a process of grouping data objects into disjointed clusters so that the data in the same cluster are similar, yet data belonging to different clusters are different. Different fuzzy data clustering algorithms exist such as Fuzzy C-Means (FCM), Possibilistic C-Means (PCM), Fuzzy Possibilistic C-Means (FPCM) and Possibilistic Fuzzy C-Means (PFCM). In this paper we present the implementation of PFCM algorithm in Matlab and we test the algorithm on two different data sets.

Index Terms— Data clustering, Clustering algorithms, K-Means, FCM, PCM, FPCM, PFCM.

I. INTRODUCTION

There are various algorithms of data clustering, every algorithm has its own advantages and disadvantages. Such as K-Means algorithm does not allow overlapping of clusters which is a major disadvantage of this algorithm. The FCM has problems dealing with noise and outliers. The PCM has the problem of coincident clusters and the FPCM has difficulties when the data set is big because the typicality values will be very small. All the apparent problem of FPCM is that it imposes a constraint on the typicality values (sum of the typicalities over all data points to a particular cluster is 1). We relax the constraint on the typicality values but retain the column constraint on the membership values. The PFCM is a good clustering algorithm to perform classification tests because it possesses capabilities to give more importance to typicalities or membership values. PFCM is a hybridization of PCM and FCM that often avoids various problems of PCM, FCM and FPCM.

II. PFCM CLUSTERING ALGORITHM

PFCM lead to optimize the following objective function:

\[
\min \left\{ J_{m,n}(U, T, V; X) \right\}
\]

subject to the constraints \( \sum_{i=1}^{n} u_{ik} = 1 \ \forall \ k, \ 0 \leq u_{ik}, t_{ik} \leq 1 \).

Here \( a > 0, \ b > 0, \ m > 1, \eta > 1 \) and \( J_{m,n} \) is the objective function. \( U \) is the partition matrix. \( T \) is the typicality matrix. \( V \) is a vector of cluster centers, \( X \) is a set of all data points, \( x \) represents a data point, \( n \) is the number of data points and \( c \) is the number of cluster centers which are described by \( s \) coordinates. \( \| x_k - v_i \|_A \) is any norm used to calculate the distance between \( i^{th} \) cluster center and \( k^{th} \) data set(also represented by \( D_{i,k} \)). Here we are using Euclidean Distance(calculated by using Equation 4).

The constants \( a \) and \( b \) define the relative importance of fuzzy membership and typicality values in the objective function. Note that, in Eq. 1 \( u_{ik} \) has the same meaning of membership as that in FCM. Similarly, \( t_{ik} \) has the same interpretation of typicality as in PCM. If we increase the importance (weight) of membership then that necessarily forces us to reduce the importance of typicality by the same amount. Also, we will see later that the optimal typicality values depend on the magnitude of \( b \). So by constraining \( a + b = 1 \), we lose modeling...
flexibility. PFCM uses the objective functions of PCM and FCM given in Eq. 2 and Eq. 3 respectively.

$$\min \left\{ f_m(U, V; X) = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m \| x_k - v_i \|^2 \right\}$$

$$\min \left\{ p_m(T; X, \gamma) = \sum_{k=1}^{n} \sum_{i=1}^{c} (t_{ik})^m \| x_k - v_i \|^2 + \sum_{i=1}^{n} \gamma_i \sum_{k=1}^{n} (1 - t_{ik})^m \right\}$$

$$D_{ikA} = \left[ \sum_{j=1}^{s} (x_{kj} - v_{ij})^2 \right]^{1/2}$$

If $b = 0$, and $\gamma_i = 0$ for all $i$, then Eq. 1 reduces to the FCM optimization problem in Eq. 2; while converts it to the usual PCM model in Eq. 3. Later, we will see that when $b = 0$, even if we do not set $\gamma_i = 0$ for all $i$, Equation 1 implicitly becomes equivalent to the FCM model. Like FPCM, under the usual conditions placed on c-means optimization problems, we get the first-order necessary conditions for extrema of $f_{m, \eta}$.

$$u_{ik} = \left( \frac{\sum_{j=1}^{s} (D_{ikA})^{2/(m-1)}}{D_{ikA}} \right)^{-1}$$

$$t_{ik} = \frac{1}{1 + \left( \frac{k}{\eta} \right)^{1/(\eta-1)}}$$

$$v_i = \frac{\sum_{k=1}^{n} (a u_{ik}^m + b t_{ik}^m) x_k}{\sum_{k=1}^{n} (a u_{ik}^m + b t_{ik}^m)}$$

PFCM behaves like FCM as the exponents grow without bound. That is, irrespective of the values of the constants $a$ and $b$, all $c$ centroids approach the overall (grand) mean as $m \to \infty$ and $\eta \to \infty$.

Equation 7 shows that if we use a high value of $b$ compared to $a$, then the centroids will be more influenced by the typicality values than the membership values. On the other hand, if we use a higher value of $a$ then the centroids will be more influenced by the membership values. Thus, to reduce the effect of outliers, we should use a higher value for $b$ than $a$. Similar effects can also be obtained by controlling the choice of $\eta$. For example, if we use a large value of $m$ and a smaller value for $\eta$, then the effect of outliers on the centroids will be reduced. However, a very large value of $m$ will reduce the effect of memberships on the prototypes and the model will behave more like the PCM model.

III. EXPERIMENTAL ANALYSIS

We run the algorithm for the two data sets. The Matlab code for following six files for Possibilistic Fuzzy C-Means clustering algorithm are as follows:

```matlab
function [center,U,OBJ_FCN]= pfcf(data,cluster_n,options)
```

```matlab
%pfcmDatasetclusteringusingpossiblistic fuzzy c-meansclustering.
```

%findsN_CLUSTER

```matlab
[CENTER,U,OBJ_FCN]=FCM(DATAN_CLUSTER)
```

%number of clusters in the data set DATA.

%DATA is size M-by-N, where M is the number of data points and N is the number of coordinates for each datapoint. The coordinates for each cluster center are returned in the rows of the matrix CENTER. The membership function of matrix U contains the grade of membership of each data point in each cluster. The values 0 and 1 indicate no membership and full membership respectively. Grades between 0 and 1 indicate that the data point has partial membership in a cluster. At each iteration, a possibilistic degree
% same as in pcm T objective function is minimized to find the best location for the
% clusters and its values are returned in OBJ_FCN.

% [CENTER..]=PFCM(DATA,N_CLUSTER,OPTIONS)
% specifies a vector of options for the
% clustering process:
% OPTIONS(1): exponent for the matrix U (default: 2.0)
% OPTIONS(2): maximum number of iterations (default: 100)
% OPTIONS(3): minimum amount of improvement (default: 1e-5)
% OPTIONS(4): info display during iteration (default: 1)
% OPTIONS(5): user defined constant a (default: 1)
% OPTIONS(6): user defined constant b should be greater than a (default: 4)
% OPTIONS(7): user defined constant nc (default: nc=2.0)

% The clustering process stops when the maximum number of iterations is reached, or when
% objective function improvement between two consecutive iterations is less than the minimum amount specified. Use NaN to select the default value. See also pinitf, tinitf, pdistfcm, pstepfcm

if nargin ~= 2 & nargin ~= 3,
    error('Too many or too few input arguments!');
end

data_n = size(data, 1);
in_n = size(data, 2);

% Change the following to set default options
default_options = [2;100;1e-5;1;1;4;2];

if nargin == 2, options = default_options;
else
    % If "options" is not fully specified, pad it with % default values.
    if length(options) < 7,
        tmp = default_options;
        tmp(1:length(options)) = options;options = tmp;
    end
end

% If some entries of "options" are NaN's, replace them with defaults.
    nan_index = find(isnan(options)==1);
    options(nan_index) = default_options(nan_index);

% The exponent should be greater than 1!
if options(1) <= 1,
    error('The exponent should be greater than 1!');
end

expo = options(1);
max_iter = options(2);
min_impro = options(3);
display = options(4);
a = options(5);
b = options(6);
nc = options(7);

ni = input('enter value of ni');
obj_fcn = zeros(max_iter, 1);

% Array for objective function
center = input('initialize the centre');
U = pinitf(cluster_n, data_n); % Initial fuzzy partition
T = tinitf(cluster_n, data_n); % Initial typicality matrix
for i = 1:max_iter,
    [U,T,center, obj_fcn(i)] = pstepfcm(data,center,U,T,cluster_n, expo,a,b,nc,ni);
if display,
    fprintf('Iteration count = %d, obj. fcn = %f
', i, obj_fcn(i));
end
% check termination condition
if i > 1,
    if abs(obj_fcn(i) - obj_fcn(i-1)) < min_impro, break;
end,end
iter_n = i; % Actual number of terations
obj_fcn(iter_n+1:max_iter) = [];

function T = pinitf(cluster_n, data_n)
T = rand(cluster_n, data_n);
col_sum = sum(T);
u=col_sum(ones(cluster_n, 1), :);
T = T./u;

function U = pinitf(cluster_n, data_n)
U = rand(cluster_n, data_n);
col_sum = sum(U);
u=col_sum(ones(cluster_n, 1), :);
U = U./u;

function [U_new,T_new,center_new,obj_fcn]= 
pstepfcm(data,center,U,T,cluster_n, expo,a,b,nc,ni)
% PSTEPFCM One step in possiblistic fuzzy c-mean clustering performs one iteration of pfcm
% clustering, where DATA: matrix of data to be
% clustered. (Each row is a data point.) U: partition
% matrix. (U(i,j) is the MF value of data j in cluster % j.) CLUSTER_N: number of clusters. EXPO:
% exponent (> 1) for the partition matrix. U_NEW:
% new partition matrix of pcm T_new: new partition
% matrix of pcm CENTER: center of clusters. (Each % row is a center.) obj_fcn: objective function for % partition U and T. Note that the situation of
% "singularity" (one of the data points in exactly the % same as one of the cluster centers)
% is not checked. However, it hardly occurs in % practice.
mf = U.^expo; % MF matrix after exponential % modification
tf=T.^nc; tfo=(1-T).^nc;
center_new = (a.*mf+b.*tf)*data./((ones(size(data, 2), 1)*sum(a.*mf'+b.*tf'))')
dist = distfcm(center, data)
% fill the distance matrix
obj_fcn=sum(sum((dist.^2).*(a.*mf+b.*tf)))+sum(ni
*sum(tfo))
% objective function
tmp = dist.^(-2/(expo-1));
U_new=tmp./(ones(cluster_n,1)*sum(tmp));
tmpt=((b/ni)*dist.^2).^(1/(nc-1));
T_new = 1./(1+tmpt);

function out = pdistfcm(center, data)
%PDISTFCM Distance measure in possiblistic fuzzy
%c-mean clustering.OUT = PDISTFCM(CENTER, %DATA) calculates the Euclidean distance between
%each row in CENTER and each row in DATA, and
%returns a distance matrix OUT of size M by N,
%where M and N are row dimensions of CENTER %and DATA, respectively, and OUT(I, J) is the
%distance between CENTER(I,:) and DATA(J,:).

out = zeros(size(center, 1), size(data, 1));
% fill the output matrix
if size(center, 2) > 1,
  for k = 1:size(center, 1),
    out(k, :) = sqrt(sum(((data-onens(size(data, 1),1)*center(k,:)).^2)));
  end
else
  for k = 1:size(center, 1),
    out(k, :) = abs(center(k)-data);
  end
end

centpfcm.m
%see also
%pfcm,dataset1,pinitf,tinitf,pdistfcm,pstepfcm.

[center,U,T,obj_fcn] = pfcm(dataset1,2);
plot(dataset1(:,1), dataset1(:,2),'o');
hold on;
maxU = max(U);
%Find data points with highest grade of membership
%in cluster1
index1 = find(U(1,:) == maxU);
line(dataset1(index1,1),dataset1(index1,2),marker,'*',
     color,'g');
line(dataset1(index1,1),dataset1(index1,2),marker,'*',
     color,'r');
%Plot the cluster centers
plot([center(1 2,1)],center(1 2,2)),'*',color,'k')
hold off;

dataset1.dat(data set-1)

The file dataset1.dat stores the whole data set on which PFCM clustering algorithm is to be applied.The result obtained after calling pfcm.m function(implemented in matlab) for the above data set-1 and data Set-2 is shown below:

OUTPUT:

Centpfcm
center value of n=0.08
initialize the centre[0.07 0.36 0.40 0.99]

Iteration count = 1, obj. fcn = 747.953767
Iteration count = 2, obj. fcn = 115.889242
Iteration count = 3, obj. fcn = 114.865973
Iteration count = 4, obj. fcn = 112.480753
Iteration count = 5, obj. fcn = 106.975976
Iteration count = 6, obj. fcn = 97.427050
Iteration count = 7, obj. fcn = 87.573956
Iteration count = 8, obj. fcn = 82.246424
Iteration count = 9, obj. fcn = 80.609839
Iteration count = 10, obj. fcn = 80.231624
Iteration count = 11, obj. fcn = 80.145442
Iteration count = 12, obj. fcn = 80.123254
Iteration count = 13, obj. fcn = 80.116857
Iteration count = 14, obj. fcn = 80.114881
Iteration count = 15, obj. fcn = 80.114251
Iteration count = 16, obj. fcn = 80.114046
Iteration count = 17, obj. fcn = 80.113980
Iteration count = 18, obj. fcn = 80.113958
Iteration count = 19, obj. fcn = 80.113951

Figure-1

fcmdata.dat(Data Set-2)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.21895919</td>
<td>0.71971108</td>
</tr>
<tr>
<td>2</td>
<td>0.67929541</td>
<td>0.31389837</td>
</tr>
<tr>
<td>3</td>
<td>0.83096535</td>
<td>0.47903993</td>
</tr>
<tr>
<td>4</td>
<td>0.103461635</td>
<td>0.77931485</td>
</tr>
<tr>
<td>5</td>
<td>0.68677271</td>
<td>0.52219568</td>
</tr>
<tr>
<td>6</td>
<td>0.091964891</td>
<td>0.71405761</td>
</tr>
<tr>
<td>7</td>
<td>0.65391896</td>
<td>0.16631787</td>
</tr>
<tr>
<td>8</td>
<td>0.70199059</td>
<td>0.422224</td>
</tr>
<tr>
<td>9</td>
<td>0.91032383</td>
<td>0.31885553</td>
</tr>
<tr>
<td>10</td>
<td>0.73068188</td>
<td>0.45760685</td>
</tr>
<tr>
<td>11</td>
<td>0.63263857</td>
<td>0.14380738</td>
</tr>
<tr>
<td>12</td>
<td>0.7226604</td>
<td>0.3206678</td>
</tr>
<tr>
<td>13</td>
<td>0.75335583</td>
<td>0.46373323</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>88</td>
<td>0.73686582</td>
<td>0.30887354</td>
</tr>
<tr>
<td>89</td>
<td>0.78386552</td>
<td>0.23765611</td>
</tr>
<tr>
<td>90</td>
<td>0.28215589</td>
<td>0.92267568</td>
</tr>
<tr>
<td>91</td>
<td>0.81972609</td>
<td>0.16874402</td>
</tr>
<tr>
<td>92</td>
<td>0.60101011</td>
<td>0.27616354</td>
</tr>
<tr>
<td>93</td>
<td>0.8235372</td>
<td>0.50918299</td>
</tr>
<tr>
<td>94</td>
<td>0.15773118</td>
<td>0.83858058</td>
</tr>
<tr>
<td>95</td>
<td>0.23359919</td>
<td>0.72674889</td>
</tr>
<tr>
<td>96</td>
<td>0.63471744</td>
<td>0.33203456</td>
</tr>
<tr>
<td>97</td>
<td>0.79476981</td>
<td>0.45126372</td>
</tr>
<tr>
<td>98</td>
<td>0.69624281</td>
<td>0.38926233</td>
</tr>
</tbody>
</table>

Centerpfcm.m

% See also
% fcadata,pfcm,pinitf,tinitf,pdistfcm,pstepfcm.
Figure 2

IV. RESULT

The result shown in figure-1 and figure-2 is quite good as compare to FCM and PCM. The PFCM algorithm overcome the problem of Overlapping of the coincident clusters.

V. CONCLUSION
We have implemented the PFCM in Matlab taking all the considerations in mind that about the membership value and typicality.

REFERENCES


