Realization of Possibilistic Fuzzy C-Means Algorithm(PFCM) in Python

Preetham Reddy.A

July 2018

Abstract

Possibilistic Fuzzy C-Means (PFCM) is clustering algorithm to cluster unlabeled data . In this document we will see how it is implemented in python and result of an example data set .

1 Introduction

If we have a dataset $X = \{x_1, ..., x_n\}$ of n dimensional data point $x_k \in R^d$.The aim of PFCM is to cluster the data and find the respective prototype(cluster center) $v_i \in R^d$,from prototypes $V = \{v_1, ..., v_c\}$ where $n >> c$.Possibilistic Fuzzy C-Means Clustering Algorithm (PFCM) was introduced by Pal et al. [1] to overcome the drawbacks of Fuzzy C-Means Clustering Algorithm (FCM)and Possibilistic C-Means Clustering Algorithm (PCM).

The cost function of a PFCM is:

$$min_{(U,V,T)}\left\{ J_{m,\eta}(U,T,V;X) = \sum_{k=1}^{n}\sum_{i=1}^{c}\left(au_{ik}^{m} + bt_{ik}^{\eta}\right)\|x_k - v_i\|_A^2 + \sum_{i=1}^{c}\gamma_i\sum_{k=1}^{n}\left(1-t_{ik}\right)^{\eta}\right\} \tag{1}$$

PFCM is an optimization problem with the constraints $\sum_{i=1}^{c} u_{ik} = 1$ and $0 \leq t_{ik}, u_{ik} \leq 1$.Here $a > 0, b > 0, m > 1, \eta > 1, \gamma > 0$ are user defined constants.a and b constants relates to the membership and typicality values in the cost function. where m is fuzzifer to the membership , η is related to the typicality,γ is values are computed using (2) which gives good results compared to user-defined values from [1].

$$\gamma_i = K\frac{\sum_{k=1}^{n} u_{ik}^{m} D_{ikA}^2}{\sum_{k=1}^{n} u_{ik}^{m}} \tag{2}$$

where $K = 1$ and $D_{ikA} = \|x_k - v_i\|_A$

Using an alternating optimization scheme membership u_{ik},typicality t_{ik} and prototypes(cluster center) v_i are updated iteratively.
Membership :

$$u_{ik} = \left(\sum_{j=1}^{c}\left(\frac{D_{ikA}}{D_{jkA}}\right)^{\frac{2}{m-1}}\right)^{-1} \tag{3}$$

where $1 < i < c$,$1 < k < n$

Typicality:

$$t_{ik} = \frac{1}{1 + \left(\frac{b}{\gamma_i} D_{ikA}^2\right)^{\frac{1}{\eta-1}}} \tag{4}$$

cluster center(prototypes):

$$v_i = \frac{\sum_{k=1}^{n} \left(a u_{ik}^m + b t_{ik}^\eta\right) x_k}{\sum_{k=1}^{n} \left(a u_{ik}^m + b t_{ik}^\eta\right)} \tag{5}$$

2 PFCM Algorithm

1)Select the unlabeled data set $X = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, ..x_n\}$
2)Select the number of clusters c $(1 < c < n)$
3)Select the constants
m $(m > 1)$,η $(\eta > 1)$,a, b $(a > 0, b > 0)$
4)Pick the norm $\|x_k - v_i\|_A$
5)Select the termination criteria
number of iteration max_t and tolerance ϵ
6)Calculate γ_i using equation (2)
7)**while** $t < max_t$ **do**
  7.1)Calculate membership u_t using equation (3)
  7.2)Calculate typicality t_t using equation (4)
  7.3)Calculate cluster centers v_t using equation (5)
  7.4)**if** $\|v_t - v_{t-1}\| < \epsilon$ **then**
    break
  end
end

Algorithm 1: PFCM algorithm

PFCM Algorithm produces three values:
A c x n fuzzy partition matrix or membership matrix U of X
A c x n possibility matrix or typicality matrix T of X
A c point prototypes $V = \{v_1, ..., v_c\}$ that are cluster centers(prototypes) of X

3 Realization in Python

Package requirements to run the PFCM algorithm :

1.Python
2.Numpy
3.Scipy

Main function of python program contain argument parsing like data,constants, number of prototypes(cluster centers),number of iteration,tolerance for convergence and calculating membership and cluster centers (prototypes) for FCM algorithm ,calculating membership , typicality and cluster centers (prototypes) for PFCM algorithm and displaying membership ,typicality and cluster centers (prototypes) of PFCM in terminal

```
def main():
    '''
    Parameters::

    data:float
    data values,data parsing through text file,rows vectors seperated by comma(,)
    c:int
    number of clusters(prototypes)(default value 2)
    e:float
    tolerance for the convergence of algorithm(default value 1e-5)
    max_iteration:int
    number of iteration for the optimization problem(default value 100)
    m:float
    fuzzifer(default value 2)
    a:float
    constant for the membership(default value 2)
    b:float
    constant for the typicallity(default value 2)
    eta:float
    constant(default value 2)
    v0:float
    selection of cluster centre random from the given data or FCM cluster center (
                                                default random)

    Attributes(displays)::

    displays membership,data,typicality,cluster center(prototypes),constants of FCM and
                                                PFCM algorithm

    '''
    # argument parsing values
    parser = argparse.ArgumentParser()
    parser.add_argument('-data',  help="data in the text file ,only numbers,numbers
                                                should be seprated by comma(,),no
                                                Default value")
    parser.add_argument('-a',  help="constant or weight,Default value is 0.5",type=float
                                                , default=0.5)
    parser.add_argument('-b',  help="constant or weight,Default value is 0.5",type=float
                                                , default=0.5)
    parser.add_argument('-m',  help="fuzzifer value,Default value is 2",type=float,
                                                default=2)
    parser.add_argument('-t',  help="epsilon or torleance value to the terminate the
                                                iterations,Default value is 1e-5",type=
                                                float, default=1e-5)
    parser.add_argument('-itera', help=" maximum iterations for clustering the data,
                                                Default value is 100",type=int, default=
                                                100)
    parser.add_argument('-eta',  help=" eta value constant ,Default value is 2",type=
                                                float, default=2)
    parser.add_argument('-c',  help="number of prototypes,Default value is 2",type=int,
                                                default=2)
    parser.add_argument('-initial', help="initial cluster centers two options first '
                                                None' for random selection,second 'fcm'
                                                for fcm clustercenter,Default value is
                                                None",type=str, default=None)
    parsed = parser.parse_args()
    #iris data with labeled data to skip the labels
    #data=np.loadtxt(parsed.data, delimiter=',',usecols=(0,1,2,3))
    #print('data',parsed.data)


    # data from the text file
    data=np.loadtxt(parsed.data, delimiter=',')
    print('data \n',data)
    # number of clusters
    print('\n no of clusters:',parsed.c)

    # Fuzzy C Mean algorithm
    # input data,no of clusters,fuzzifer,tolerance,iteration,metric,intial cluster
                                                center
    # output  final membership,final cluster centers and fcm_equidistance
    fcm_cluster_center_final, fcm_membership_final,fcm_equidistance =fcm(data,parsed.c,
```

```
                                                parsed.m, parsed.t, parsed.itera, metric
                                                ="euclidean", v0=None)
  # Fuzzy C Mean final cluster centers
    print("fcm final centres \n",fcm_cluster_center_final)
    #print('FCM membership \n',fcm_membership_final.T)
    # weight constants (a,b) print warning if greater than one

    if parsed.a+parsed.b>1:
        print('\n warning a+b>1')
        print('\n a:',parsed.a)
        print('\n b:',parsed.b)
        print('\n a+b:',parsed.a+parsed.b)

    else:
        print('\n a:',parsed.a)
        print('\n b:',parsed.b)

    #print('\n fuzzifer:',parsed.m)
    #print('\n eta:',parsed.eta)
    # gamma values are calculated using Fuzzy C Mean cluster centers
    #membership u,distance d,fuzzifer m default value 2
    gamma1 = gamma_value(fcm_membership_final, fcm_equidistance, parsed.m)
    print('\n Gamma:',gamma1)
     # PFCM intial cluster can be random initialization from data or Fuzzy C Mean
                                                    cluster centers(prototypes)
    if parsed.initial=='fcm' :
       pfcm_cluster_center_initial=fcm_cluster_center_final

    else:
       pfcm_cluster_center_initial=None

    # Possibilistic Fuzzy C Mean (PFCM) Algorithm function call
    # input data,no of clusters,fuzzifer,tolerance,iteration,metric,intial cluster,
                                                weight constants(a,b),gamma,eta
    # output intial and final (membership and cluster centers),typicallity
    pfcm_cluster_center_final,  pfcm_membership_final,pfcm_typicality_final=pfcm(data,
                                                parsed.c, parsed.m, parsed.t,parsed.
                                                itera,a=parsed.a,b=parsed.b,eta=parsed.
                                                eta,gamma=gamma1, metric="euclidean", v0
                                                =pfcm_cluster_center_initial)
    # prints data,final cluster,membership,typicality

    print('PFCM final cluster centre \n',pfcm_cluster_center_final)
    print('PFCM membership \n',pfcm_membership_final.T)
    print('PFCM typicality \n',pfcm_typicality_final.T)



    return 0
```

FCM algorithm is used for calculating the γ value for PFCM using FCM membership values and euclidean distance between cluster centres(Prototypes) and data points.

```
def fcm(data, cluster, fuzzifer, tolerance, max_iterations, metric="euclidean", v0=None)
                                                :
    '''
    Parameters::
    data:float
    data values
    cluster:int
    number of prototypes(clusters)
    tolerance:float
    tolerance for the convergence of alogrithm
    max_iteration:int
    number of iteration for the optimization problem
    metric:float
    metric norm is equlidean distance
    v0:float
    random number selection of cluster centre from the given data

    Attributes::
```

```
    u:float
    final membership values
    V:float
    final prototypes(cluster centres)
    d:float
    euclidean distance between cluster centre and data point
    '''


    ## check the length of data
    if not data.any() or len(data) < 1 or len(data[0]) < 1:
        print("Error: Data is in incorrect format")
        return

    # size  of  Features, Datapoints in data
    S, N = data.shape
    ## check the number of cluster
    if not cluster or cluster <= 0:
        print("Error: Number of clusters must be at least 1")
    ## check the Fuzzifer value
    if not fuzzifer:
        print("Error: Fuzzifier must be greater than 1")
        return

    # Initialize the cluster centers

    if v0 is None:
        # Pick random values from dataset

        v0 = data[np.random.choice(data.shape[0], int(cluster), replace=False), :]
        print('fcm intial centres \n',v0)
    # List of all cluster centers (Bookkeeping)
    v = np.empty((max_iterations, int(cluster), N))

    v[0] = np.array(v0)

    # Membership Matrix Each Data Point in eah cluster
    u = np.zeros((max_iterations, int(cluster), S))

    # Number of Iterations
    t = 0

    while t < max_iterations - 1:
        # upadting membership values
        u[t], d = membership(data, v[t], fuzzifer, metric)
        # upadting cluster centers(prototypes) values

        um,v[t + 1] = update_clusters(data, u[t], fuzzifer)

        costfunction=um*d

        # Stopping Criteria
        if np.linalg.norm(v[t + 1] - v[t]) < tolerance:

            break

        t += 1


    return v[t], u[t - 1],d
```

FCM algorithm is an iterative optimization problem we need to update membership value and cluster centers(Prototypes)

```
def membership(x, v, m, metric):
    '''
    Parameters::
    x:float
    data values
    v:float
    prototypes(cluster centres) values
```

```
    m:float
    fuzzifer value

    metric:float
    metric norm is equlidean distance




    Attributes::
    d:float
    euclidean distance between cluster centre and data point
    u:float
    membership values
    '''

    # calculating equidistance
    d = cdist(x, v, metric=metric).T

    # Sanitize equidistances (Avoid Zeroes)
    d = np.fmax(d, np.finfo(np.float64).eps)
    # power
    exp = -2. / (m - 1)
    d2 = d ** exp

    u = d2 / np.sum(d2, axis=0, keepdims=1)

    return u, d

def update_clusters(x, u, m):
    '''
    Parameters::
    x:float
    data points

    u:float
    membership value of fcm

    m:float
    fuzzifer value

    Attributes::
    v:float
    cluster centres(prototypes)  of FCM
    um:float
    membership value power to fuzzifer for FCM costfunction
    '''
    um = u ** m

    v = um.dot(x) / np.atleast_2d(um.sum(axis=1)).T

    return um, v
```

Now we have the FCM final cluster centers(prototypes) and membership values. so, we can calculate the γ values using equation (2) for the PFCM using FCM membership values and euclidean distance between FCM final cluster centres(prototypes) and data points

```
def gamma_value(u, d, m):

    '''
    calculation of gamma values using equation(2) from[1] for PFCM

    Parameters::
    u:float
    membership value of fcm
    d:float
    equlidiean distance between data and prototype
    m:float
    fuzzifer value
```

```
    Attributes::
    n:float
    gamma values
    '''
    u = u ** m
    # aviod zeros
    u = np.fmax(u, np.finfo(np.float64).eps)
    d2=d**2
    n = np.sum(u * d2, axis=1) / np.sum(u, axis=1)

    return n
```

PFCM Algorithm to find cluster centers(prototypes) ,membership ,typicality values.

```
def pfcm(data, cluster, fuzzifer, tolerance, max_iterations,a,b,eta,gamma, metric="
                                                euclidean", v0=None):
    '''
    Parameters::
    data:float
    data points
    cluster:int
    number of clusters(prototypes)
    fuzzifer:float
    fuzzifer value
    tolerance :float
    tolerance for the convergence of alogrithm
    max_iteration:int
    number of iteration for the optimization problem
    a:float
    weight(constant) for membership in PFCM
    b:float
    weight(constant) for typicality in PFCM
    eta:float
    user defined constant for typicality
    gamma:float
    calculated using FCM cluster centers and distance between prototypes and data points
    metric:float
    metric norm is equlidean distance
    v0:float
    random number selection of cluster centre from the given data or FCM cluster center,
                                                    default is random from the given data


    Attributes::
    u[t - 1]:float
    final membership values of PFCM
    typ[t]:float
    final typicality values of PFCM
    v[t]:float
    final cluster centres values of PFCM

    '''

    ## check the length of data
    if not data.any() or len(data) < 1 or len(data[0]) < 1:
        print("Error: Data is in incorrect format")
        return

    # Num Features, Datapoints
    S, N = data.shape
    ## check the number of cluster
    if not cluster or cluster <= 0:
        print("Error: Number of clusters must be at least 1")
    ## check the fuzzifer value
    if not fuzzifer:
        print("Error: Fuzzifier must be greater than 1")
        return

    # Initialize the cluster centers random from data  if FCM cluster is not given

    if v0 is None:
```

```
        v0 = data[np.random.choice(data.shape[0],int(cluster), replace=False), :]
        print(' pfcm intial centres \n',v0)
    else:

        print(' pfcm intial centres \n',v0)
    #  cluster centers
    v = np.empty((max_iterations, int(cluster), N))

    v[0] = np.array(v0)

    # Membership matrix Each Data Point in eah cluster
    u = np.zeros((max_iterations, int(cluster), S))


    t = 0
    # typicallity matrix Each Data Point in eah cluster
    typ= np.zeros((max_iterations,int(cluster), S))
    while t < max_iterations - 1:
        # upadting membership values
        u[t], d = membership(data, v[t], fuzzifer, metric)
        # upadting typicality values
        typ[t],d=typicality(data, v[t], gamma, fuzzifer, metric,b,eta)
        # upadting cluster centers(prototypes) values
        um,v[t + 1],k = update_pfcmclusters(data, u[t],typ[t], fuzzifer,a,b,eta)

        penality=(1-typ[t])**eta

        costfunction=k*(d.T**2)+gamma.sum()*penality

        # Stopping Criteria
        if np.linalg.norm(v[t + 1] - v[t]) < tolerance:

            break

        t += 1
        if t==max_iterations - 1:
           typ[t]=typ[t-1]

    return v[t],  u[t - 1],  typ[t]
```

PFCM is also an iterative optimization problem we need to update the membership values using equation (3) ,typicality values using equation (4) and cluster centers (Prototypes) using equation (5) using alternative optimization scheme.

```
def update_pfcmclusters(x, u,typ, m,a,b,eta):
    '''
    Paratmeters::
    x:float
    data point values

    u:float
    membership value of PFCM
    typ:float
    typicality value of PFCM

    m:float
    fuzzifer value
    a:float
    constant for the memberships
    b:float
    constant for the typicallity
    eta:float
    user-defined constant for typucality


    Attributes::
    v:float
    prototypes(cluster centres) values for PFCM
    um:float
    membership values power to fuzzifer  for PFCM costfunction
```

```
    k:float
    membership values and typicallity constant mulitplication for PFCM costfunction
    '''
    um = u ** m
    typeta = typ ** eta
    aum=a*um

    btypeta=b*typeta
    k=aum+btypeta

    v=k.dot(x)/np.atleast_2d(k.sum(axis=1)).T

    return um, v,k

def membership(x, v, m, metric):
    '''
    Parameters::
    x:float
    data values
    v:float
    prototypes(cluster centres) values
    m:float
    fuzzifer value

    metric:float
    metric norm is equlidean distance




    Attributes::
    d:float
    euclidean distance between cluster centre and data point
    u:float
    membership values
    '''

    # calculating equidistance
    d = cdist(x, v, metric=metric).T

    # Sanitize equidistances (Avoid Zeroes)
    d = np.fmax(d, np.finfo(np.float64).eps)
    # power
    exp = -2. / (m - 1)
    d2 = d ** exp

    u = d2 / np.sum(d2, axis=0, keepdims=1)

    return u, d

def typicality(x, v, gamma, m, metric,b,eta):
    '''
    Paraments::
    x:float
    data values
    v:float
    prototypes(cluster centres) values of PFCM
    gamma:float
    gamma values
    m:float
    fuzzifer value
    metric:float
    metric norm is equlidean distance
    b:float
    constant for the typicallity
    eta:float
    user-defined constant for typucality


    Attributes::
    t:float
    Typicallity values of PFCM
    d:float
```

```
    euclidean distance between cluster centre and data point
    '''
    d = cdist(x, v, metric=metric)

    d = np.fmax(d, np.finfo(np.float64).eps)
    #d[d==0]=1
    d1 = b / gamma

    power=(d ** 2)

    d2=d1*power

    exp = 1. / (eta - 1)

    d2 = d2 ** exp

    addone=(1. + d2)

    #typicallity
    t= 1. / addone

    return t.T,d
```

	data[0]	data[1]
0	-5.00	0.00
1	-3.34	1.67
2	-3.34	0.00
3	-3.34	-1.67
4	-1.67	0.00
5	1.67	0.00
6	3.34	1.67
7	3.34	0.00
8	3.34	-1.67
9	5.00	0.00
10	0.00	0.00
11	0.00	10.00

Figure 1: Two dimensional artificial dataset

4 Results

To see the how the PFCM algorithm performs a two dimensional artificial dataset is taken from [1].figure.1 is the data set and figure.2 its plot on two dimensional axis and there results are based on different user-defined constant values explained through examples 1,2,3.

Note: Matplotlib package is used to visualize the data but it is not included in the source code.

The blue X marks represents data point and orange dots marks represents cluster centers(prototypes) in the plot figures .

Example 1: $a = 0$,$b = 0.5$,$m = 2$,$\eta = 2$,$c = 2$,γ calculated using equation (2), initial cluster centers are randomly initialized from given dataset. When $a = 0$ the cost function of PFCM converts to cost function of PCM and PCM has a drawback of coincident cluster we can see coincident cluster in the plot figure 3. membership values of x_{11}, x_{12} are similar, but typicality values of x_{11}, x_{12} are different and

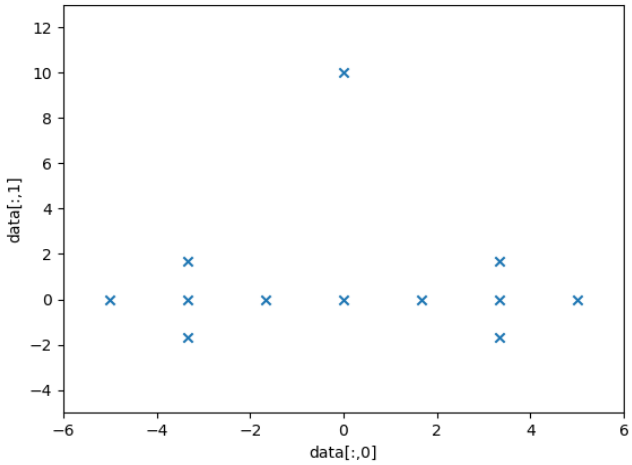

Figure 2: Plot of two dimensional artificial dataset.

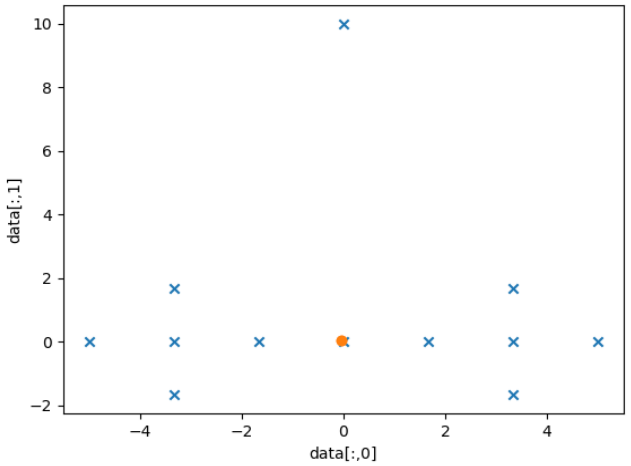

Figure 3: Plot of dataset and PFCM final cluster center for the example 1 .

cluster centers,membership,typicality of PFCM are shown in figure.4.

Example 2: $a = 0.5$,$b = 0$,$m = 2$,$\eta = 2$,$c = 2$,γ calculated using equation (2), initial cluster centers are randomly initialized from given dataset. When $b = 0$ the cost function of PFCM converts to equivalent cost function of FCM . since b is equal to zero typicality values of PFCM becomes one and cluster centers(prototypes) of PFCM are equal to FCM cluster centers(prototypes) results are shown in figure.5.

Example 3: $a = 0.5$,$b = 0.5$,$m = 20$,$\eta = 20$,$c = 2$,γ calculated using equation (2), initial cluster centers are randomly initialized from given dataset. when $m \to \infty$,$\eta \to \infty$ the values of typicality will lead to 0.5 ,the values of membership will lead to $\frac{1}{c}$ and results are shown in figure.6.

Note: for the example 4 and 5 we use iris dataset .

Example 4: $a = 0.5$,$b = 0.5$,$m = 2$,$\eta = 2$,$c = 3$,γ calculated using equation (2) , initial cluster centers are randomly initialized from given iris dataset. Results are shown in figure.7.PFCM cluster centers v_1, v_2 are similar. cluster centers of v_1, v_2 are coincident cluster centers to avoid the those is-

```
fcm intial centres
 [[ 5.    0.  ]
 [-3.34  1.67]]
fcm final centres
 [[ 2.98537934  0.54352326]
 [-2.98536306  0.54353206]]

 Gamma: [ 7.88136578  7.88144132]
 pfcm intial centres
 [[-3.34  1.67]
 [ 3.34  1.67]]
PFCM final cluster centre
 [[-0.04589307  0.04538096]
 [ 0.0458822   0.04538148]]
PFCM membership
 [[ 0.50947344  0.49052656]
 [ 0.51146848  0.48853152]
 [ 0.51417878  0.48582122]
 [ 0.51122198  0.48877802]
 [ 0.52832483  0.47167517]
 [ 0.47167537  0.52832463]
 [ 0.48853148  0.51146852]
 [ 0.48582127  0.51417873]
 [ 0.48877811  0.51122189]
 [ 0.49052658  0.50947342]
 [ 0.49994168  0.50005832]
 [ 0.49999997  0.50000003]]
PFCM typicality
 [[ 0.39120052  0.38221496]
 [ 0.53901733  0.52760064]
 [ 0.59244732  0.57868342]
 [ 0.53348708  0.52230102]
 [ 0.85677804  0.84229075]
 [ 0.84228777  0.85677754]
 [ 0.52759689  0.53901839]
 [ 0.57867949  0.59244801]
 [ 0.52229735  0.53348812]
 [ 0.38221166  0.39120174]
 [ 0.999727    0.99972707]
 [ 0.13723499  0.13723614]]
```

Figure 4: cluster center,membership,typicality of PFCM and cluster center of FCM for the example 1.

```
fcm intial centres
 [[ 3.34  1.67]
 [ 1.67  0.  ]]
fcm final centres
 [[ 2.98537453  0.5435305 ]
 [-2.98536787  0.54352482]]

 Gamma: [ 7.88141415  7.88139295]
 pfcm intial centres
 [[-3.34  1.67]
 [ 0.    0.  ]]
PFCM final cluster centre
 [[-2.98537533  0.54353058]
 [ 2.98536707  0.54352474]]
PFCM membership
 [[ 0.93635781  0.06364219]
 [ 0.96731783  0.03268217]
 [ 0.98965828  0.01034172]
 [ 0.89935988  0.10064012]
 [ 0.91557418  0.08442582]
 [ 0.08442169  0.91557831]
 [ 0.03268328  0.96731672]
 [ 0.01034159  0.98965841]
 [ 0.10063821  0.89936179]
 [ 0.06364291  0.93635709]
 [ 0.49999543  0.50000457]
 [ 0.50000049  0.49999951]]
PFCM typicality
 [[ 1.  1.]
 [ 1.  1.]
 [ 1.  1.]
 [ 1.  1.]
 [ 1.  1.]
 [ 1.  1.]
 [ 1.  1.]
 [ 1.  1.]
 [ 1.  1.]
 [ 1.  1.]
 [ 1.  1.]]
```

Figure 5: cluster center,membership,typicality of PFCM and cluster center of FCM for the example 2.

```
fcm intial centres
 [[ 0.    0.  ]
 [ 3.34  1.67]]
fcm final centres
 [[-2.80589628  0.48334455]
 [ 2.80589623  0.48334467]]

 Gamma: [ 7.56967996  7.56968055]
 pfcm intial centres
 [[  0.    10.  ]
 [ -1.67   0.  ]]
PFCM final cluster centre
 [[ 1.51884303  0.31804492]
 [-1.51884302  0.31804492]]
PFCM membership
 [[ 0.48357534  0.51642466]
 [ 0.47898269  0.52101731]
 [ 0.47453679  0.52546321]
 [ 0.48247069  0.51752931]
 [ 0.44214653  0.55785347]
 [ 0.55785347  0.44214653]
 [ 0.52101731  0.47898269]
 [ 0.52546321  0.47453679]
 [ 0.51752931  0.48247069]
 [ 0.51642466  0.48357534]
 [ 0.5         0.5       ]
 [ 0.5         0.5       ]]
PFCM typicality
 [[ 0.48639187  0.50281913]
 [ 0.49317315  0.51419868]
 [ 0.49409793  0.51957303]
 [ 0.49211786  0.50965259]
 [ 0.50510618  0.56288581]
 [ 0.5628858   0.50510618]
 [ 0.51419868  0.49317315]
 [ 0.51957303  0.49409793]
 [ 0.50965259  0.49211786]
 [ 0.50281913  0.48639187]
 [ 0.52417175  0.52417175]
 [ 0.47570939  0.47570939]]
```

Figure 6: cluster center,membership,typicality of PFCM and cluster center of FCM for the example 3.

```
fcm intial centres
 [[ 5.7  2.8  4.5  1.3]
 [ 6.1  2.8  4.7  1.2]
 [ 5.1  3.8  1.5  0.3]]
fcm final centres
 [[ 5.88919095  2.76123177  4.36424221  1.39743989]
 [ 6.7751083   3.0524278   5.64690111  2.05360338]
 [ 5.00356134  3.40303609  1.48500085  0.25154075]]

 a: 0.5

 b: 0.5

 Gamma: [ 0.5823638   0.68940677  0.34456563]
 pfcm intial centres
 [[ 5.8  4.   1.2  0.2]
 [ 5.2  4.1  1.5  0.1]
 [ 6.1  3.   4.6  1.4]]
PFCM final cluster centre
 [[ 5.0284693   3.36309757  1.5926328   0.29100946]
 [ 5.0409242   3.36537187  1.61553046  0.30186835]
 [ 6.34658525  2.91533662  5.03589852  1.74778651]]
```

Figure 7: cluster centers of PFCM and cluster center of FCM for the example 4.

```
fcm intial centres
 [[ 6.8  3.2  5.9  2.3]
 [ 7.7  2.8  6.7  2. ]
 [ 4.8  3.4  1.6  0.2]]
fcm final centres
 [[ 5.88920666  2.76123742  4.36426517  1.39745172]
 [ 6.7751273   3.05243333  5.64692478  2.0536125 ]
 [ 5.00356139  3.40303534  1.48500212  0.25154133]]

 a: 0.5

 b: 0.5

 Gamma: [ 0.58236445  0.68940334  0.3445687 ]
 pfcm intial centres
 [[ 5.88920666  2.76123742  4.36426517  1.39745172]
 [ 6.7751273   3.05243333  5.64692478  2.0536125 ]
 [ 5.00356139  3.40303534  1.48500212  0.25154133]]
PFCM final cluster centre
 [[ 5.98236062  2.81012195  4.47598633  1.45089474]
 [ 6.50208025  2.97382854  5.27923507  1.89414929]
 [ 5.00458726  3.39449776  1.49566221  0.2534977 ]]
```

Figure 8: cluster centers of PFCM and cluster center of FCM for the example 5.

sues FCM final cluster centers can be used as PFCM initial cluster centers explained in the next example.

Example 5: $a = 0.5$,$b = 0.5$,$m = 2$,$\eta = 2$,$c = 3$,γ calculated using equation (2), initial cluster center are taken from FCM final cluster center . Results are shown in figure.8.

5.References

[1] Nikhil R. Pal , Kuhu Pal,James M.keller and James C.Bezdek
A Possibilistic Fuzzy c-Means Clustering Algorithm

[2] Scipy and Numpy Documentation
https://docs.scipy.org/doc/

[3] Python Documentation
https://docs.python.org/3/index.html