Fuzzy C-Means Clustering for Image Segmentation

Saad Alam

Pattern Recognition
University of Cincinnati

1 Introduction

Image segmentation is a key aspect of image analysis and image interpretation. It plays a vital role in the fields of robot vision, medical imaging, and object recognition. The primary goal of segmenting an image is to divide the image into partitions composed of image sections that share similar attributes such as color, tone, texture, etc. There are several image segmentation methodologies which can be categorized into four major types: thresholding, clustering, edge detection, and region extraction. For this particular project, a clustering approach will be taken to solve the problem of image segmentation.

A cluster within a data set can be described as a group of points with similar features around a center point, i.e. a centroid. Clusters can be characterized with definite boundaries or without definite boundaries. Clustering methods that define clusters without definite boundaries are called fuzzy clustering methods. In fuzzy clustering, a given data point can have membership in more than one cluster of a data set. In this project, a fuzzy clustering algorithm is considered. The clustering algorithm takes a set of unlabeled data, which in the application of image segmentation consists of the image pixel values, and determines the membership of each point to each cluster in the image.

The Fuzzy C-Means (FCM) clustering algorithm was originally proposed by J.C. Dunn in his paper *A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters* published in 1973. The FCM uses an iterative approach to calculate the membership of each data point to each cluster. The number of clusters is decided by the user. Several variants of the FCM algorithm have been developed for various applications.

For the purposes of image segmentation, one of the striking shortcomings of the FCM algorithm is that it fails to take into account spatial information of the pixels. For many images, pixels located spatially far from each other tend not be similar and should belong to the same segment of an image. To address this specific possibility for improvement, this project explored one variant of the FCM algorithm known as the Fuzzy C-Means with Spatial Constraints (FCMS) algorithm.

2 Objectives of the Project

The main objective of the project was to first understand and implement the FCM algorithm and then proceed to implement the FCMS which is an improved version of the algorithm for image segmentation applications. The following steps were taken to meet the objective:

- Understand and implement the FCM algorithm
- Understand and implement the FCMS algorithm
- Compare the results of the two algorithms

3 Related Theory

3.1 Original Fuzzy C-Means Clustering Algorithm

The Fuzzy C-Means clustering algorithm (FCM) is based on minimizing an objective function. The objective function is given in equation (1) below:

$$J_{FCM} = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m d^2(x_k, v_i) \qquad (1)$$

Where $X = \{x_1, x_2, \dots, x_n\}$ is the data set of interest, n is the number of data items, c is the number of clusters, u_{ik} is the membership of the data point x_k in the i^{th} cluster, m is a weighting exponent on each fuzzy membership, v_i is the center of cluster i, and $d^2(x_k, v_i)$ is the Euclidean distance squared between data point x_k and cluster center v_i. The above objective function must be minimized under the following constraints:

$$\sum_{i=1}^{c} u_{ik} = 1, \forall k \qquad (2a)$$

$$u_{ik} \in [0,1] \qquad (2b)$$

$$0 < \sum_{k=1}^{n} u_{ik} < n, \forall i \qquad (2c)$$

The constraint specified by equation (2a) states that for any given data point x_k, its membership across all the clusters must sum to 1. Equation (2b) says that the membership values must be between 0 and 1. Lastly, Equation (2c) states that the sum of memberships to one cluster across all the points must be less than the number of data points.

The input to the FCM algorithm is the set of data points $X = \{x_1, x_2, \dots, x_n\}$, the number of clusters c that the data should be segmented into, a fuzziness parameter m, and a threshold ϵ used to determine when to stop the iterations. The main output of the algorithm is a matrix $U = [u_{ik}]$ which is a $n\ x\ c$ matrix which specifies the membership of each data point to each cluster. The point $U(i,k)$, written as u_{ik}, of the U matrix specifies the membership of data point x_k to cluster v_i. Another output of the algorithm is $V = \{v_1, v_2, \dots, v_i\}$ which is the set of cluster centroids.

A numerical solution to minimize the objective function can be obtained with the following iteration process:

1. Set the values for c, m, and ϵ
2. Initialize the membership matrix $U = \{u_{ik}\}$ with random values under the constraint specified by equation (2)
3. Calculate the cluster centers using the values in the U matrix:

$$v_i = \frac{\sum_{k=1}^{n} (u_{ik})^m x_k}{\sum_{k=1}^{n} (u_{ik})^m} \qquad (3)$$

4. Update the membership matrix U:

$$u_{ik} = \frac{1}{\sum_{j=1}^{c} \left(\frac{d^2(x_i, v_k)}{d^2(x_j, v_k)} \right)^{\frac{1}{m-1}}} \qquad (4)$$

5. If $\left| |U_{new} - U_{old}| \right| < \epsilon$, stop; otherwise go to step 3.

3.2 Fuzzy C-Means Clustering with Spatial Constraints

A modification to the FCM algorithm has been proposed to take into account spatial constraints of the image. The additional information serves to make the algorithm results more robust and accurate, especially in the presence of noise. This is accomplished by modifying the objective function of the original FCM algorithm:

$$J_{FCMS} = \sum_{k=1}^{n} \sum_{i=1}^{c} (u_{ik})^m d^2(x_k, v_i) + \frac{\alpha}{N_R} \sum_{i=1}^{c} \sum_{k=1}^{n} (u_{ik})^m \sum_{r \in N_k} d^2(x_r, v_i) \qquad (5)$$

where N_k specifies all the points that fall into a neighborhood around the point x_k, N_R is the cardinality of the neighborhood, and the parameter α is a control parameter that adjusts the effect of the additional term. The additional term is what takes into account spatial information of the image pixels.

The optimization problem is minimizing the objective function J_{FCMS} specified in equation (5) under the following constraints:

$$u_{ik} \in [0,1] \qquad (6)$$

$$\sum_{i=1}^{c} u_{ik} = 1, \forall k \qquad (7)$$

$$0 < \sum_{k=1}^{n} u_{ik} < n, \forall i \qquad (8)$$

The constrained optimization problem can be solved using the method of Lagrange Multipliers. A penalty term is added to equation (5) as follows:

$$F_m = \sum_{i=1}^{c} \sum_{k=1}^{N} \left((u_{ik})^m d^2(x_k, v_i) + \frac{\alpha}{N_R} (u_{ik})^m \gamma_i \right) + \theta \left(1 - \sum_{i=1}^{c} u_{ik} \right) \qquad (9)$$

where $\gamma_i = \sum_{r \in N_k} d^2(x_r, v_i)$. Taking the derivative of F_m with respect to u_{ik} and setting it equal to 0 yields the following result

$$\left[\frac{\delta F_m}{\delta u_{ik}} = m(u_{ik})^{m-1} d^2(x_k, v_i) + \frac{\alpha m}{N_R} (u_{ik})^{m-1} \gamma_i - \theta \right]_{u_{ik} = u_{ik}^*} = 0 \qquad (10)$$

Solving for u_{ik}^* gives the following:

$$u_{ik}^{*} = \left(\frac{\theta}{m\left(d^2(x_k, v_i) + \frac{\alpha}{N_R}\gamma_i\right)} \right)^{\frac{1}{m-1}} \qquad (11)$$

Using the constraint of equation (7) gives:

$$\sum_{j=1}^{c} \left(\frac{\theta}{m\left(d^2(x_k, v_j) + \frac{\alpha}{N_R}\gamma_j\right)} \right)^{\frac{1}{m-1}} = 1 \qquad (12)$$

Which can be rearranged as the following:

$$\theta = \frac{m}{\sum_{j=1}^{c} \left(\frac{d^2(x_k, v_i) + \frac{\alpha}{N_R}\gamma_i}{d^2(x_k, v_j) + \frac{\alpha}{N_R}\gamma_j} \right)^{\frac{1}{m-1}}} \qquad (13)$$

Substituting equation (11) into equation (13) yields the solution for u_{ik}^{*}:

$$u_{ik}^{*} = \frac{\left(d^2(x_k, v_i) + \frac{\alpha}{N_R}\sum_{r \in N_k} d^2(x_r, v_i)\right)^{-\frac{1}{m-1}}}{\sum_{j=1}^{c}\left(d^2(x_k, v_j) + \frac{\alpha}{N_R}\sum_{r \in N_k} d^2(x_r, v_j)\right)^{-\frac{1}{m-1}}} \qquad (14)$$

The next step is to find an optimum value of the cluster centroids. This time the derivative of F_m in (9) is taken with respect to v_i yielding the following result:

$$\left[\sum_{k=1}^{n} u_{ik}^{m} d(x_k, v_i) + \sum_{k=1}^{n} u_{ik}^{m} \left(\frac{\alpha}{N_R}\right) \sum_{x_r \in N_k} d(x_r, v_i) \right]_{v_i = v_i^{*}} = 0 \qquad (15)$$

Solving for v_i yields the following result:

$$v_i^{*} = \frac{\sum_{k=1}^{n} (u_{ik})^m (x_k + \frac{\alpha}{N_R}\sum_{r \in N_k} x_r)}{(1+\alpha)\sum_{k=1}^{n} (u_{ik})^m} \qquad (16)$$

The iteration steps of the FCMS algorithm remain nearly identical to those of the FCM algorithm. The steps of the modified algorithm are given below:

1. Set the values for c, m, ϵ, and α
2. Initialize the membership matrix $U = \{u_{ik}\}$ with random values under the constraint specified by equation (2)
3. Calculate the cluster centers using the values in the U matrix using equation (16).

4. Update the membership matrix U using equation (14).
5. If $\left|\left|V_{new} - V_{old}\right|\right| < \epsilon$, stop; otherwise go to step 3.

4 Experimental Results

The image segmentation performance of the FCM algorithm was compared to that of the FCMS algorithm to quantify the improvement in the results. For all the experiments, unless otherwise specified, the penalty parameter α was set to 3 for the FCMS algorithm. The neighborhood of a given pixel for the experiments consists of all the pixels that fall within a $3\ x\ 3$ window of the pixel excluding itself. The fuzziness index m was set to 2. The first image on which the algorithms were tested was a synthetic image composed of two rectangles and additive Gaussian noise with a mean of 0 and 10% variance. The number of clusters c was set to 2 for both algorithms for the first input image. The first input image is shown below in Figure 1

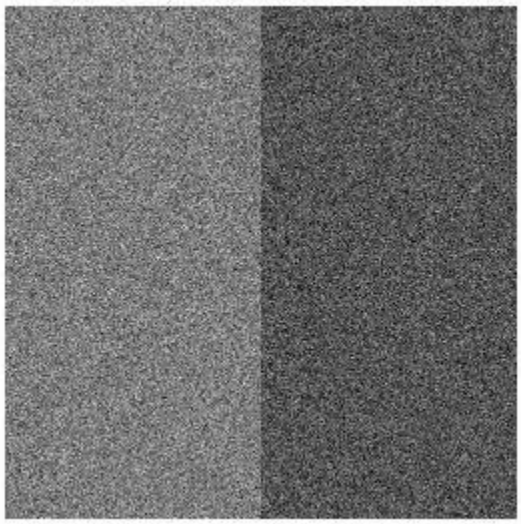

Figure 1 Input Image 1

The image was first segmented using the original FCM algorithm. The result is shown in Figure 2.

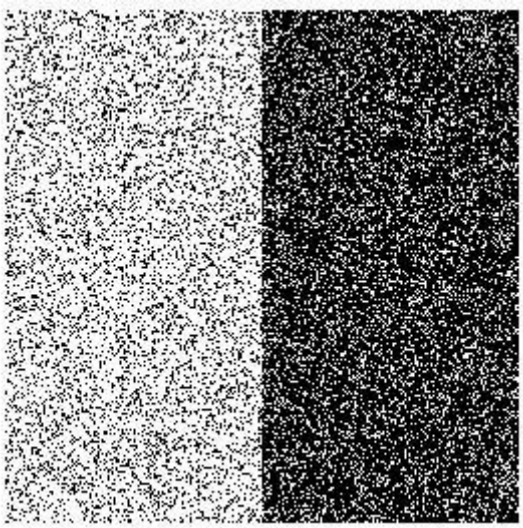

Figure 2 FCM Segmentation

The image was then segmented using the FCMS algorithm. The result is shown below in Figure 3.

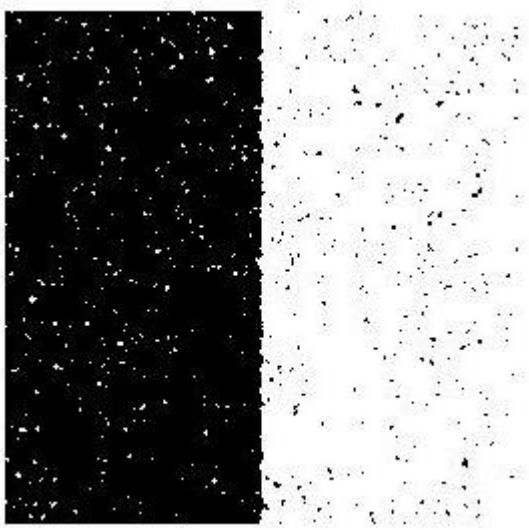

Figure 3 FCMS Segmentation

It is evident from comparing Figures 2 and 3 above that the FCMS clustering algorithm performed significantly better than the original FCM algorithm. In fact, the accuracy of the FCMS algorithm was 98.16% for the input image in Figure 1 as oppose to the FCM algorithm accuracy of 78.67%.

After the two algorithms were run on a synthetic image with added noise, they were run on an actual photograph. The input image is shown in Figure 4.

Figure 4 Input Image 2

The second image that is segmented is an actual photograph. It has much more detail than the first synthetic image. This image, however, is free of noise. The image was segmented using the two algorithms. This time, however, the number of clusters c was designated as 4. The result from the original FCM algorithm is shown in Figure 5.

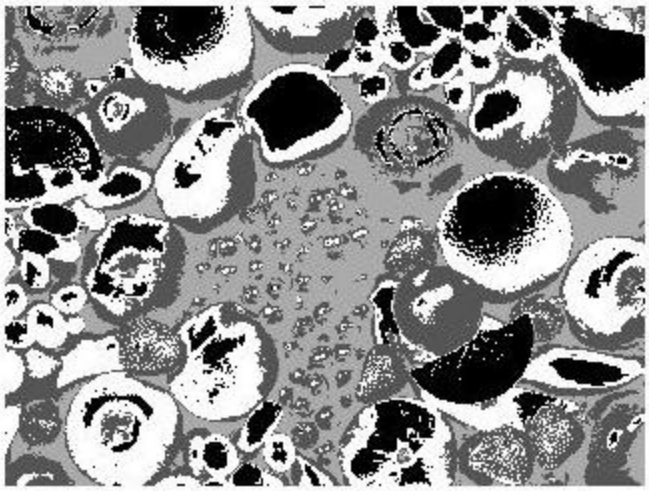

Figure 5 FCM Segmentation

Next, the image was segmented using the FCMS algorithm. The result is shown in Figure 6.

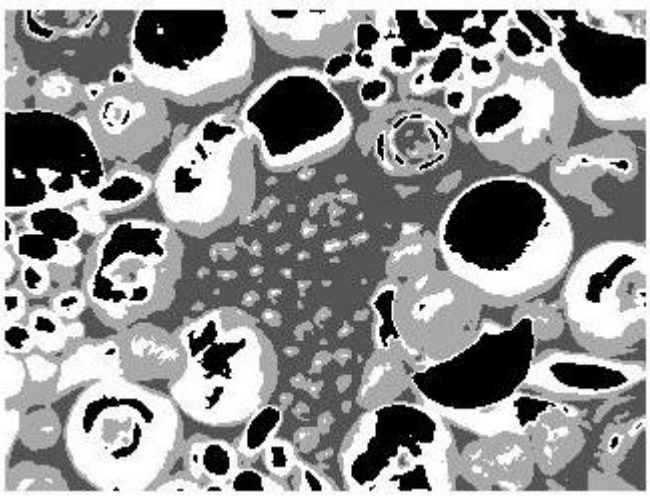

Figure 6 FCMS Segmentation

Comparing the two segmented images reveals that the FCMS algorithm yields much more crisp and uniform results. This is expected as the FCMS takes into account the spatial location of each pixel. As a result, the pixels closer to each other are much more likely to be placed in the same cluster. In the FCM algorithm, however, there is no consideration given to the spatial location of the pixel. As a result, there is more segment diversity within regions of the image. Note that the conclusions drawn are not based solely on the two images presented in the paper, but also on other images. The two input images presented in the report are two representative sample experiments.

Once the segmentation results had been compared for the images, the FCMS algorithm was executed on the first input image for varying values of alpha in order to quantify the effect that the value had on the segmentation results. Figure 7 shows a graph shows the accuracy of the image segmentation of the input image in Figure 1 versus the value of alpha.

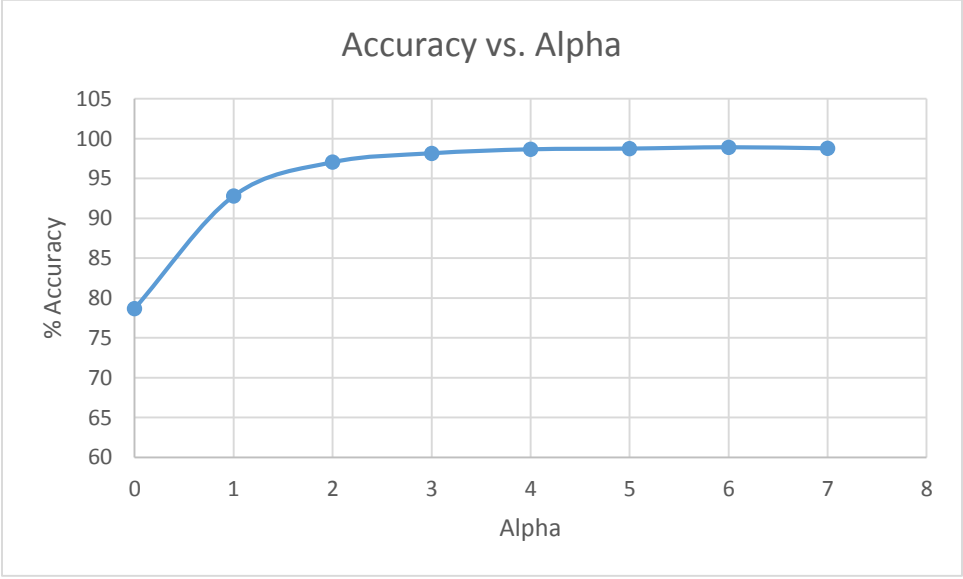

Figure 7 FCMS Accuracy vs. Alpha

The graph in Figure 7 seems to show that the accuracy of the algorithm reaches a maximum at α value of 3.

5 Conclusions

In this report, the FCM and FCMS algorithms for image segmentation have been explained and demonstrated on sample images. The clustering algorithms are based upon minimizing an objective function. Specifically for the FCMS algorithm, the objective function was obtained by modifying the objective function of the original FCM algorithm to allow the clustering of a specific pixel to be influenced by the clustering of pixels in a neighborhood of that pixel. The neighborhood biases the clustering of each pixel towards the labels of the pixels that are spatially close to it.

Both algorithms were performed on a grayscale synthetic image as well as a grayscale photograph. In both cases the FCMS algorithm produced better results than the FCM algorithm. It should be noted, however, that the FCMS algorithm significantly increases the amount of computation required for the clustering process. As the number of clusters and image resolution increases the required amount of computation increases drastically. This problem requires further study and optimization to make the clustering method more computationally feasible for image segmentation.

The results presented in this report are a preliminary study and evaluation of Fuzzy C-Means Algorithms. One further method of improving the performance of the algorithms is to use kernel distance functions as oppose to simple Euclidean distance. Also, it may be that spatial location information may not be important for a certain image segmentation application and therefore the computational expense for using the FCMS is unnecessary. Overall, the FCMS algorithm provides one method of improving the FCM clustering algorithm for image segmentation by additional utilizing spatial information of the image.

Works Cited

M. N. Ahmed, S. M. Yamany, N. Mohamed, A. A. Farag, and T. Moriarty, "A Modified Fuzzy C-Means Algorithm for Bias Field Estimation and Segmentation of MRI Data", *IEEE Transactions On Medical Imaging,* vol. 21, No.3, March 2002

Y. Yang, and S. Huang, "Image Segmentation By Fuzzy C-Means Clustering Algorithm with a Novel Penalty Term", *Computing and Informatics,* vol. 26, pp. 17-31, 2007

S. Chen and D. Zhang, "Robust Image Segmentation Using FCM With Spatial Constraints Based on New Kernel-Induced Distance Measure", *IEEE Transactions On Systems, Man, and Cybernetics—Part B: Cybernetics,* vol. 34, No. 4, August 2004

J. Bezdek, R. Ehrlich, and W. Full, "FCM: The Fuzzy C-Means Clustering Algorithm", *Computers & Geosciences,* vol. 10, No. 2-3, pp. 191-203, 1984.

APPENDIX – MATLAB Code

**Note, the Matlab m files have been included in the zip file. They were modified for each of the experiments accordingly.

```
I = zeros(256, 256);
I(:, 1:128) = 100;
I(:, 129:256) = 60;
X = imnoise(mat2gray(I, [0 255]), 'gaussian', 0.1);
%X = rgb2gray(imread('tropicalfruits_small.jpg'));
figure
imshow(X);
[row col dim] = size(X);

%Defining the number of data points, clusters, and dimension of points
num_data_points = row*col;
num_clusters = 2;
num_dimensions = dim;
alpha = 7;
neighbor_dist = 2;
num_neighbors = 8;

%Minimum and maximum value that each data point dimension can take
minimum = 0;
maximum = 255;

%Initialize degree of membership matrix
degree_of_membership = initialize_dom(num_data_points, num_clusters);

%Set C-Means control parameters
epsilon = 0.001;
fuzziness = 2;
max_diff = 1;

%Matrix of all data points
[data_point spatial_info] = initialize_data(X, row, col, dim);

%Find neighbors of all pixels
weights = zeros(num_data_points, num_neighbors);
temp = zeros(num_data_points,2);
for k = 1:num_data_points
    temp(:,1) = abs(spatial_info(k,1) - spatial_info(:,1));
    temp(:,2) = abs(spatial_info(k,2) - spatial_info(:,2));
    w = (temp(:,1) <= 1) & (temp(:,2) <= 1);
    w(k) = 0;
    neighbors = find(w);
    weights(k,:) = [neighbors' zeros(1, num_neighbors - length(neighbors))];
end
clear temp;
clear w;

means = zeros(num_data_points, num_dimensions);
for k = 1:num_data_points
   a = weights(k,:);
   a = a(a ~= 0);
```

```
   means(k,:) = mean(data_point(a), 1);
end

centroids = calculate_centroids(degree_of_membership, data_point,
num_clusters, num_dimensions, fuzziness, means, alpha);
while (max_diff > epsilon)
    %Calculate cluster centers
    centroids_old = centroids;

    %Calculate distance of each point from each cluster centroid
    distances = calculate_dist(num_data_points, num_clusters, num_dimensions,
data_point, centroids);

    %Calculate new degrees of membership
    degree_of_membership_new = calculate_dom(distances, num_clusters,
num_data_points, fuzziness, alpha, weights);

    %max_diff = max(max(abs(degree_of_membership -
degree_of_membership_new)))

    degree_of_membership = degree_of_membership_new;
    centroids = calculate_centroids(degree_of_membership, data_point,
num_clusters, num_dimensions, fuzziness, means, alpha);
    max_diff = max(max(abs(centroids_old - centroids)))
end

[~, C] = max(degree_of_membership, [], 2);
y = linspace(0, 255, num_clusters);

Output = X;
for i = 1:num_data_points
    Output(spatial_info(i,1), spatial_info(i,2)) = y(C(i));
end
figure
imshow(mat2gray(Output));
```

```
function degree_of_mem = initialize_dom( num_points, num_clusters )
%Initialize a random probability for each member of the degree of
%membership matrix
degree_of_mem = zeros(num_points, num_clusters);
for i = 1:num_points
    s = 0; %Sum of probability
    r = 100; %Remaining probability
    for j = 2:num_clusters
        rval = randi([0 r], 1, 1);
        r = r - rval;
        degree_of_mem(i,j) = rval/100;
        s = s + degree_of_mem(i,j);
    end
    if (s > 1)
        s = 1;
    end;
    degree_of_mem(i,1) = 1 - s;
end

end


function [initial spatial] = initialize_data(I, num_rows, num_cols, num_dim)
%Initializes the matrix of data points to be used in the clustering
%algorithm
initial = zeros(num_rows*num_cols, num_dim);
spatial = zeros(num_rows*num_cols, 2);
k = 1;
for i = 1:num_rows
    for j = 1:num_cols
       initial(k, :) = I(i, j, :);
       spatial(k, :) = [i j];
       k = k+1;
    end
end


end


function distances = calculate_dist(num_points, num_clusters, num_dim,
data_points, centroids)
%Calculates the new degree of membership of each point
distances = zeros(num_points, num_clusters);
temp = zeros(num_points, num_dim);
for i = 1:num_clusters
    for j = 1:num_dim
        temp(:,j) = data_points(:,j) - centroids(i,j);
    end
    distances(:,i) = sqrt(sum(temp.^2, 2));
end

end
```

```
function centroids = calculate_centroids(deg_membership, data_points,
num_clusters, num_dimensions, q, means, alpha)
%Calculate centroids of clusters
centroids = zeros(num_clusters, num_dimensions);
mem = deg_membership.^q;
temp = data_points + alpha*means;
for i = 1:num_clusters
    numerator = (mem(:,i)')*(temp);
    denominator = (1+alpha)*sum(mem(:,i), 1);
    centroids(i, :) = (1/denominator)*numerator;
end

end


function deg_membership = calculate_dom(distances, num_clusters, num_points,
q, alpha, weights)
%Calculates the new degree of membership of each point
deg_membership = zeros(num_points, num_clusters);
for k = 1:num_points
    a = weights(k, :);
    a = a(a ~= 0);
    for i = 1:num_clusters
        b = distances(:,i);
        b = b(a);
        deg_membership(k,i) = (distances(k,i)^2 +
(alpha/length(b))*sum(b.^2))^(-1/(q-1));
    end
    deg_membership(k,:) = deg_membership(k,:)./(ones(1,
num_clusters)*sum(deg_membership(k,:),2));
end

end
```