Veronica S. Moertini

INTRODUCTION TO FIVE DATA CLUSTERING ALGORITHMS

Abstract

Clustering is one of the primary tasks of data mining. Over the years, many methods have been developed for clustering patterns. Each method can have its own technique (i.e. partitioning or hierarchical), mode (on-line or off-line), approach (fuzzy or crisp clustering), or special purpose (i.e. for sequential data set, very large database, etc.). This paper aims to introduce the most representative algorithms used in off-line mode that apply crisp or fuzzy approach. The algorithms are K-Means, Fuzzy C-Means (FCM), Mountain, Subtractive and psFCM. The implementation of two of the algorithms using Matlab is provided in the appendix.

Abstraksi

Clustering merupakan salah satu dari tugas utama data mining. Beberapa tahun terakhir ini, banyak metodologi telah berhasil dikembangkan untuk melakukan proses clustering pada pola-pola yang dikehendaki. Setiap metodologi clustering dapat memiliki teknik, mode, pendekatan ataupun tujuan khusus tertentu. Makalah ini bertujuan untuk memperkenalkan lima algoritma clustering yang banyak digunakan, yaitu K-Means, Fuzzy C-Means (FCM), Mountain, Subtractive and psFCM. Implementasi dari dua algoritma tersebut diberikan di Appendix.

Diterima : 1 Agustus 2002
Disetujui untuk dipublikasikan : 15 Agustus 2002

1. Introduction

Clustering is one of the primary tasks of data mining[1]. An example of clustering applications in a knowledge discovery context is discovering homogeneous sub-populations for customers in marketing database. The discovery can then be used to design effective marketing strategies.

Over the years, many methods have been developed for clustering patterns. Each method can have its own technique (i.e. partitioning or hierarchical), approach (fuzzy or crisp clustering), or special purpose (i.e. for sequential data set, very large database, etc.). Researchers have been working to improve them, or to invent new methods to satisfy current demand. This paper aims to introduce two of the early invented clustering algorithms, which are K-Means and Fuzzy C-Means (FCM) algorithms, and three other algorithms that improve them, which are Mountain, Subtractive and psFCM clustering methods. In the appendix, the implementation of three of the algorithms using Matlab will also be provided.

2. Data Clustering

Clustering is a process of partitioning or grouping a given set of unlabeled patterns into a number of clusters such that similar patterns are assigned to one cluster. Each pattern can be represented by a vector having many parameters or attributes. Fundamental to the use of any clustering technique is the computation of a measure of *similarity* or *distance* between the respective patterns[2]. With patterns having metric properties, a *distance-type* measure can be used, whereas with patterns having qualitative components, a *matching-type* measure is appropriate. Euclidean or Mahalanhois distance can be used in measuring distances between patterns. Euclidean distance is not scale invariant. Therefore, it is suggested that data be standardized (by dividing each variable by its standard deviation) before computing the Euclidean distance. Mahalanhois distance has the advantage of explicitly accounting for any correlation that might exist between the variables. Matching-type measurement is based on the reasoning that two patterns should be viewed as being similar to the extent that they share common attributes.

Once one has settled on the choice of an appropriate similarity measure, the next step is to select a particular type of computational algorithm. Two of the popular kinds of clustering techniques used in algorithms are hierarchical technique and partitioning technique. The hierarchical technique clusters the clusters themselves at various levels, whereas the partitioning technique forms clusters by optimizing some specific clustering criterion.

There are two main approaches to clustering. One method is crisp clustering (or hard clustering), and the other one is fuzzy clustering. A characteristic of the crisp clustering method is that the boundary between clusters is fully defined. However, in many real cases, the boundaries between clusters cannot be clearly defined. Some patterns may belong to more than one cluster. In such cases, the fuzzy clustering method provides a better and more useful method to classify these patterns.

There are two modes of data clustering algorithms, which are on-line and off-line (batch) algorithm[1]. In an on-line algorithm, patterns are presented in sequence and the results of clustering are adjusted at each presentation of a new pattern. On-line clustering algorithms can be realized by unsupervised learning neural networks. In a batch algorithm, conversely, all data entries that represent patterns are available at once. Hence, the clusters are also computed at once.

This paper discusses the most representative algorithms used in batch mode that apply Euclidean distance in measuring similarities, partitioning technique, as well as crisp and fuzzy approach.

3. K-Means Clustering

The K-means algorithm, which is a crisp clustering algorithm, partitions a collection of *n* vector $\mathbf{x}_j$, $j = 1,\ldots,n$ into *c* groups G_i, $i = 1,\ldots,c$, and finds a cluster center in each group such that a cost function (or an objection function) of dissimilarity (or distance) measure is minimized[3]. When the Euclidean distance (please see Appendix) is chosen as the dissimilarity measure between a vector $\mathbf{x}_k$ in group *j* and the corresponding cluster center $\mathbf{c}_i$, the cost function can be defined by

$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c}\left(\sum_{k, x_k \in G_i} \| x_k - c_i \|^2 \right) \quad (3.1)$$

where $J_i = \sum_{k, x_k \in G_i} \| x_k - c_i \|^2$ is the cost function within group *i*. Thus, the value of J_i depends on the geometrical properties of G_i and the location of $\mathbf{c}_i$.

A generic distance function $d(\mathbf{x}_k, \mathbf{c}_i)$ can be applied for vector $\mathbf{x}_k$ in group *i*; the overall cost function can be written as

$$J = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c}\left(\sum_{k,x_k \in G_i} d(x_k - c_i)\right) \quad (3.2)$$

For simplicity, the Euclidean distance is used as the dissimilarity measure and the overall cost function is expressed in Equation (3.1).

The partitioned groups are typically defined by an c x n binary **membership matrix U**, where element u_{ij} is 1 if the j th data point x_j belongs to group i, and 0 otherwise. Once the cluster centers $\mathbf{c}_i$ are fixed, the minimizing u_{ij} for Equation (3.1) can be derived as follows

$$u_{ij} = \begin{cases} 1 & \text{if } \| x_j - c_i \|^2 \le \| x_j - c_k \|^2 \text{, for each } k \neq i, \\ 0 & \text{otherwise} \end{cases} \quad (3.3)$$

Equation (3.3) states that $\mathbf{x}_j$ belongs to group i if $\mathbf{c}_i$ is the closest center among all centers. Since a data point can only be in a group, the membership matrix **U** has the properties as follows

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1,\dots,n$$

and

$$\sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij} = n$$

If $\mathbf{u}_{ij}$ is fixed, then the optimal center $\mathbf{c}_i$ that minimize Equation (3.1) is the mean of all vectors in group i:

$$c_i = \frac{1}{|G_i|} \sum_{k,x_k \in G_i} x_k \quad (3.4)$$

where $|G_i|$ is the size of G_i, or $|G_i| = \sum_{j=1}^{n} u_{ij}$

For a batch mode operation, the K-means algorithm is presented with a data set $\mathbf{x}_i$, $i = 1,\dots,n$; the algorithm determines the cluster center $\mathbf{c}_i$ and the membership matrix **U** iteratively using the following steps:

Step 1: Initialize the cluster center $\mathbf{c}_i$, $i = 1,\dots,c$. This is typically achieved by randomly selecting c points from among all of the data points.

Step 2: Determine the membership matrix **U** by Equation (3.3).

Step 3: Compute the cost function (or objection function) by Equation (3.1). Stop if either it is below a certain tolerance value or its improvement over previous iteration is below a certain threshold.

Step 4: Update the cluster center by Equation (3.4). Go to step 2.

The algorithm is inherently iterative, and no guarantee can be made that it will converge to an optimum solution. The performance of the K-means algorithm depends on the initial positions of the cluster centers. An example of K-means algorithm implementation using Matlab can be found in the appendix.

4. Fuzzy C-Means Clustering

Fuzzy C-Means clustering (FCM), also known as ISODATA, is a data clustering algorithm in which each data point belongs to a cluster to a degree specified by a membership grade[3]. FCM and its derivatives have been used very successfully in many applications, such as pattern recognition, classification, data mining, and image segmentation[4].

FCM partitions a collection of n vector $\mathbf{x}_i$, $i = 1,\ldots,n$ into *c fuzzy* groups, and finds a cluster center in each group such that a cost function of dissimilarity measure is minimized. FCM employs fuzzy partitioning such that a given data point can belong to several groups with the degree of belongingness specified by membership grades between 0 and 1. To accommodate the introduction of fuzzy partitioning, the **membership matrix U** is allowed to have elements with values between 0 and 1. Imposing normalization stipulates that the summation of degrees of belongingness for a data set always be equal to unity:

$$\sum_{i=1}^{c} u_{ij} = 1, \forall_j = 1,\ldots,n. \qquad (4.1)$$

The cost function (or objective function) for FCM is then a generalization of Equation (3.1):

$$J(U, c_1, \ldots, c_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{j}^{n} u_{ij}^m d_{ij}^2 \qquad (4.2)$$

where u_{ij} is between 0 and 1; c_i is the cluster center of fuzzy group i; $d_{ij} = \|\mathbf{c}_i - \mathbf{x}_j\|$ is the Euclidean distance between ith cluster center and jth data point; and $m \ \varepsilon \ [1,\infty)$ is a weighting exponent.

The necessary conditions for Equation (4.2) to reach a minimum can be found by forming a new objective function J as follows:

$$\begin{aligned} J(U, c_1, \ldots, c_c.\lambda_1, \ldots, \lambda_n) &= J(U, c_1, \ldots, c_c) + \sum_{j=1}^{n} \lambda_j \left(\sum_{i=1}^{c} u_{ij} - 1\right) \\ &= \sum_{i=1}^{c} \sum_{j}^{n} u_{ij}^m d_{ij}^2 + \sum_{j=1}^{n} lj \left(\sum_{i=1}^{c} u_{ij} - 1\right) \qquad (4.3) \end{aligned}$$

where λ_j, $j = 1$ to n, are the Lagrange multipliers for the n constraints in Equation (4.1). By differentiating $J(U, c_1, \ldots, c_c, l_1, \ldots, l_n)$ to all its inputs arguments, the necessary conditions for Equation (4.2) to reach its minimum are

$$c_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m} \qquad (4.4)$$

and

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{kj}}\right)^{2/(m-1)}} \qquad (4.5)$$

The fuzzy C-means algorithm is simply an iterated procedure through the preceding two conditions. In a batch-mode operation, FCM determines the cluster centers c_i and the membership matrix **U** using the following steps:

Step 1: Initialize the membership matrix **U** with random values between 0 and 1 such that the constraints in Equation (4.1) are satisfied.

Step 2: Calculates c fuzzy cluster center c_i, $i = 1,\ldots c.$, using Equation (4.4).

Step 3: Compute the cost function (or objection function) by Equation (4.2). Stop if either it is below a certain tolerance value or its improvement over previous iteration is below a certain threshold.

Step 4: Compute a new **U** by Equation (4.5). Go to step 2.

The cluster centers can also be first initialized and then the iterative procedure carried out. No guarantee ensures that FCM converges to an optimum solution. The performance depends on the initial cluster centers, thereby another fast algorithm to determine the initial cluster centers can be used. Or, FCM is run several times, each starting with a different set of initial cluster centers. (Please see Appendix D for the implementation of FCM algorithm using Matlab.)

5. Mountain Clustering Method

The mountain clustering method is a relatively simple and effective approach to approximate estimation of cluster centers on the basis of a density measure called the **mountain function**[3]. This method can be used to obtain initial cluster centers that are required by more sophisticated cluster algorithm, such as FCM. It can also be used as a quick stand-alone method for approximate clustering. The method is based on what a human does in visually forming cluster of a data set. This method determines the cluster centers using three steps.

The first step involves forming a grid the data space, where the intersections of the grid lines constitute the candidates for cluster centers, denoted as a set *V*. A finer gridding increases the number of potential clustering centers, but it also increases the computation required. The gridding is generally evenly spaced, but it is not a requirement.

The second step involves constructing a mountain function representing a data density measure. The height of the mountain function at a point $\mathbf{v} \in V$ is equal to

$$m(v) = \sum_{i=1}^{N} \exp\left(-\frac{\| v - x_i \|^2}{2\sigma^2}\right) \qquad (5.1)$$

where $\mathbf{x}_i$ is the ith data point and σ is an application-specific constant. Equation (5.1) implies that each data point $\mathbf{x}_i$ contributes to the height of the mountain function at **v**, and the contribution is inversely proportional to the distance between $\mathbf{x}_i$ and **v**. The mountain function can be viewed as a measure of *data density* since it tends to be higher if more data points are located nearby, and lower if fewer data points are around. The constant σ determines the height as well as the smoothness of the resultant mountain function. The clustering results are normally insensitive to the value of σ, as long as the data set is of sufficient size and is well clustered.

The third step involves selecting the cluster centers by sequentially destructing the mountain function. The candidate centers *V* that has the greatest value for the mountain function need to be found. This becomes the first cluster center $\mathbf{c}_1$. (If there are more than one maxima, one of them is randomly selected as the first cluster center.) Obtaining the next cluster center requires eliminating the effect of the just-identified center, which is typically surrounded by a number of grid points that also have high density scores. This is realized by revising the mountain function; a new mountain is formed by subtracting a scaled Gaussian function at $\mathbf{c}_1$:

$$m_{new} = (v) = m(v) - m(c_1)\exp\left(-\frac{\| v - c_1 \|^2}{2\beta^2}\right) \qquad (5.2)$$

After subtraction, the second cluster center is again selected as the point in *V* that has the largest value for the new mountain function. This process of revising the mountain function and finding the next cluster centers continues until a sufficient number of cluster centers is attained.

6. Subtractive Clustering

The mountain clustering method is relatively simple and effective. However, its computation grows exponentially with the dimension of the patterns because the method must evaluate the mountain function over all grid points. For example, a clustering pattern with four variables and each dimension having a resolution of 10 grid lines would result in 10^4 grid points that must be evaluated. Subtractive clustering is an alternative approach that can be used to eliminate this problem. In subtractive clustering, data points (not grid points) are considered as the candidates for cluster centers. By using this method, the computation is simply proportional to the number of data points and independent of the dimension problem as mentioned previously[3].

Consider a collection of n data points $\{\mathbf{x}_1,\ldots,\mathbf{x}_n\}$ in an M-dimensional space. The data points are assumed to have been normalized within a hypercube. Since each data point is a candidate for cluster centers, a *density measure* at data point $\mathbf{x}_i$ is defined as

$$D_i = \sum_{j=1}^{n} \exp\left(-\frac{\| x_i - x_j \|^2}{(r_a/2)^2} \right)$$

where r_a is a positive constant. Therefore, a data point will have a high density value if it has many neighboring data points. The radius r_a defines a neighborhood; data points outside this radius contribute only slightly to the density measure.

After the density measure of each data points has been calculated, the data point with the highest density measure is selected as the first cluster center. Let $\mathbf{x}_{c1}$ be the point selected and D_{c1} its density measure. The density measure for each data point $\mathbf{x}_i$ is revised by the formula

$$D_i = D_i - D_{c1} \exp\left(-\frac{\| x_i - x_{c1} \|^2}{(r_b/2)^2} \right),$$

where r_b is a positive constant. Therefore, the data points near the first cluster center $\mathbf{x}_{c1}$ will have significantly reduced density measure, thereby making the points unlikely to be selected as the next cluster center. The constant r_b defines a neighborhood that has measurable reductions in density measure. The constant r_b is normally larger than r_a to prevent closely spaced cluster centers; generally r_b is equal to $1.5r_a$.

After the density measure for each data point is revised, the next cluster center $\mathbf{x}_{c2}$ is selected and all of the density measures for data points are revised again. This process is repeated until a sufficient number of cluster centers are generated.

7. Partition Simplification Fuzzy C-Means Clustering (psFCM)

It can be concluded from the discussion of Fuzzy C-Means Clustering that if a good set of initial cluster centers is chosen, the algorithm may take less iterations to find the actual cluster centers.

The main idea of the psFCM algorithm is to refine the initial cluster centers. It finds a set of initial cluster centers that is very close to the actual cluster centers of the dataset. Therefore, it will reduce the number of iterations required to find the actual cluster centers of the dataset. Due to space limitation, this section will simply discuss the main principle of the psFCM. (A detailed discussion of psFCM can be found in [4].)

The psFCM is divided into two phases. In *Phase I*, first, the dataset is partitioned into small block sells using the *k-d tree* method and reduce the original dataset into a *simplified dataset*. All patterns in a unit block are replaced by the centroid of these patterns. Then, the large number of patterns in the original dataset is drastically reduced to small of unit blocks' centroids, i.e., the simplified dataset. Secondly, FCM algorithm is applied to this simplified dataset to find the actual cluster centers. (Please see Figure 1 and Figure 2.) In *Phase II*, it is a standard process of the FCM with the cluster centers initialized by the final cluster centers from Phase I. It has been proved that the execution performance of the psFCM is much better than that of the FCM and its derivatives.

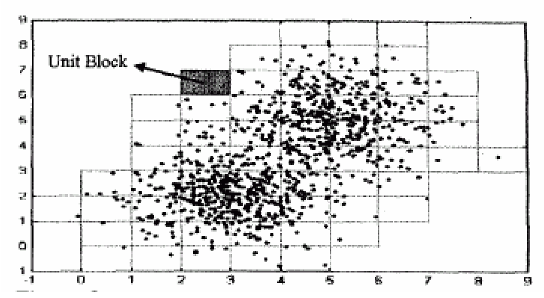

Figure 1. Partitioning the original dataset into several unit blocks^4.

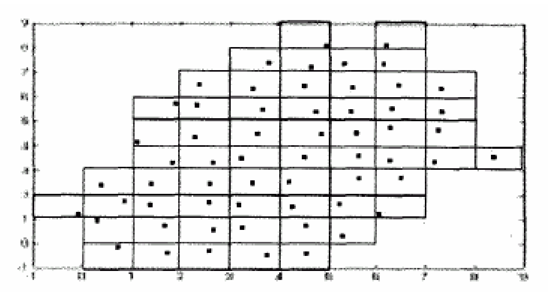

Figure 2. The simplified dataset of Figure 1^4.

8. Conclusion

K-Means clustering algorithm is inherently iterative, and no guarantee can be made that it will converge to an optimum solution. The performance of the K-means algorithm depends on the initial positions of the cluster centers.

Since the number of iterations required in the FCM algorithm strongly depends on the initial cluster centers, choosing a good set of initial cluster centers is very important for FCM. The psFCM algorithm, which has the goal of refining the initial cluster centers (selecting the initial cluster centers that is very close to the actual cluster centers of the dataset), reduces the number of iterations required in the FCM algorithm. Therefore, it improves its performance.

The mountain clustering method is a relatively simple and effective approach to approximate estimation of cluster centers on the basis of a density measure. However, its computation grows exponentially with the dimension of the patterns as the method must evaluate the mountain function over all grid points. Subtractive clustering is an alternative approach that can be used to eliminate this problem.

9. References

[1] Fayyad, U.M. et.all, "*Advances in Knowledge Discovery and Data Mining*", The MIT Press, 1996.

[2] Dillon, W.R. and Goldstein,M., "*Mutlivariate Analysis, Methods and Applications*", John Willey & Sons, 1982.

[3] Jang, J.-S.R.; Sun, C.-T, Mizutani E., "*Neuro-Fuzzy and Soft Computing*", Prentice Hall Inc., USA, 1997.

[4] Hung, MC; Yang D-L, An Efficient Fuzzy C-Means Clustering Algorithm, "*Proceedings of 2001 IEEE International Conference on Data Mining*", San Jose, California, December 2001.

[5] Zurada, J.M, "*Introduction to Artificial Neural Systems*", West Publishing Co., Singapore, 1992.

[6] The Mathworks Inc, "*Matlab Manual*", USA, 2001.

[7] The Mathworks Inc, "*Fuzzy Logic Toolbox Manual*", USA, 2001.
[8] ftp://ftp.mathworks.com/pub/books/jang/.
[9] Guralnik, V.; Karypis G., A Scalable Algorithm for Clustering Sequential Data, "*Proceedings of 2001 IEEE International Conference on Data Mining*", San Jose, California, December 2001.
[4] Sanches-Diaz G., Ruiz-Shulcloper J., A Clustering Method for Very Large Mixed Data Sets, "*Proceedings of 2001 IEEE International Conference on Data Mining*", San Jose, California, December 2001.

10. Author

Veronica Sri Moertini adalah dosen Jurusan Ilmu Komputer, FMIPA Universitas Katolik Parahyangan, Bandung.

APPENDIX

A. The distance definition in Euclidean Space

The distance between two points $P_1(x_1,y_1,z_1)$ and $P_2(x_2,y_2,z_2)$ is equal to the length of vector $\mathbf{x}_1$-$\mathbf{x}_2$[5]:

$$\| \mathbf{x}_1 - \mathbf{x}_2 \| = \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2 + (z_1-z_2)^2} \quad \text{where} \quad \mathbf{x}_1 \equiv \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \quad \text{and} \quad \mathbf{x}_2 \equiv \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$$

B. Basic definition of fuzzy set

If X is a collection of objects denoted generically by x, then a fuzzy A in X is defined as a set of ordered pairs[3]:

$$A = \{(x, m_A(x)) \mid x \in X\},$$

where $\mu_A(x)$ is called the **membership function** (or **MF**) for the fuzzy set A. The MF maps each element of X to a membership grade (or membership value) between 0 and 1. Usually X is referred to as the **universe of discourse**, or simply the **universe**, and it may consist of discrete (ordered or nonordered) objects or continuous space. Figure 3 gives the examples of MF for discrete and continuous universe.

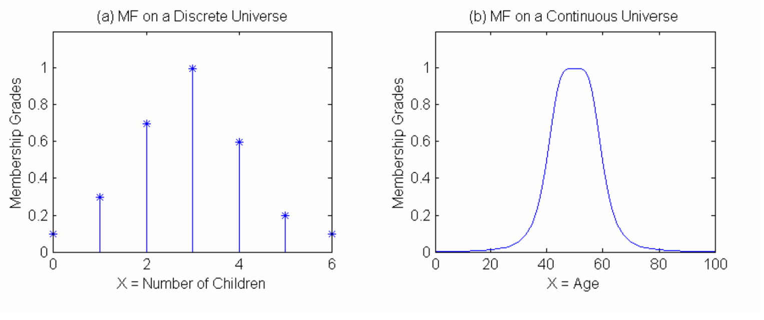

Figure 3. (a) A = "sensible number of children in a family"; (b) B = "about 50 years old".

C. Implementation of K-Means Alogrithm

The one step function of K-Means clustering algorithm is given in this section. The K-Means data clustering function, which calls this one-step function, is similar to FCM function, which can be found in Matlab's Fuzzy Toolbox.

```
function [U_new, center_new, obj_fcn] = step_kmeans(data, U, center)
% One step KMeans clustering function. Please see function stepfcm
% in Appendix D for complete reference of the parameters.
% By: Veronica S. Moertini. Created on June 2002.

% Get the number of cluster centers
cluster_n = size(center,1);

% Update the membership matrix, U
for i = 1:cluster_n,
   for j = 1:size(data,1),
       for k = 1:cluster_n,
            if k ~= i,
               if ec_dist(data(j,:), center(i,:)) <= ec_dist(data(j,:),
center(k,:)),
                    U(i,j) = 1;
                end
            end
       end
   end
end
U_new = U;

% Calculate the objective function
obj_fcn = 0;
for i = 1:cluster_n,
    for j = 1:size(data,1),
        if U(i,j) == 1,
            obj_fcn = obj_fcn + ec_dist(data(j,:), center(i,:));
        end
    end
end

% Calculate the new cluster centers
center_new = zeros(size(center,1),size(center,2));
for i = 1:cluster_n,
    gi = sum(U(i,:));
    s = zeros(1, size(center,2));
    for j = 1:size(data,1),
        if U(i,j) == 1, s = s + data(j,:); end
    end
    center_new(i,:) = s/gi;
end
```

D. Implementation of Fuzzy C-Means Alogrithm

The Matlab function of one step fuzzy C-Means algorithm is provided in this section. The function *fcm* that calls this one step function can be found in Matlab's Fuzzy Toolbox.

```
function [U_new, center, obj_fcn] = stepfcm(data, U, cluster_n, expo)
%STEPFCM One step in fuzzy c-mean clustering.
%   [U_NEW, CENTER, ERR] = STEPFCM(DATA, U, CLUSTER_N, EXPO)
%   performs one iteration of fuzzy c-mean 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.
%   CENTER: center of clusters. (Each row is a center.)
%   ERR: objective function for partition U.
%   Note that the situation of "singularity" (one of the data points is
%   exactly the same as one of the cluster centers) is not checked.
%   However, it hardly occurs in practice.
%
%   By: Roger Jang, Copyright 1994-2001 The MathWorks, Inc.

mf = U.^expo;        % MF matrix after exponential modification
center = mf*data./((ones(size(data, 2), 1)*sum(mf'))'); % new center
dist = distfcm(center, data);        % fill the distance matrix
obj_fcn = sum(sum((dist.^2).*mf));  % objective function
tmp = dist.^(-2/(expo-1));      % calculate new U, suppose expo != 1
U_new = tmp./(ones(cluster_n, 1)*sum(tmp));
```

E. Implementation of Mountain Clustering Method

The Matlab function of Mountain Clustering methods and an example of using the function can be found at: `ftp://ftp.mathworks.com/pub/books/jang/mount1.m` and `ftp://ftp.mathworks.com/pub/books/jang/mount2.m`. Combined and modified, the result of the execution of `mount1.m` and `mount2.m` is shown on Figure 4.

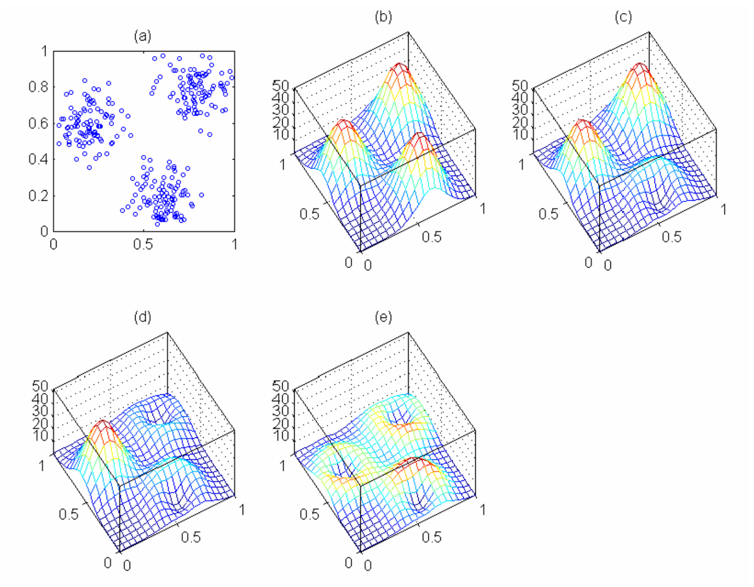

Figure 4. (a) 2-D dataset; (b) the original mountain function of the dataset with $\sigma = 0.1$; (c) mountain function after the first reduction; (d) mountain function after the second reduction; (e) mountain function after the third reduction.