An Efficient Fuzzy C-Means Clustering Algorithm

Ming-Chuan Hung and Don-Lin Yang
Department of Information Engineering, Feng Chia University
100 Wenhwa Rd., Taichung, Taiwan 407
E-mail: mchong@fcu.edu.tw, dlyang@fcu.edu.tw

Abstract

The Fuzzy C-Means (FCM) algorithm is commonly used for clustering. The performance of the FCM algorithm depends on the selection of the initial cluster center and/or the initial membership value. If a good initial cluster center that is close to the actual final cluster center can be found, the FCM algorithm will converge very quickly and the processing time can be drastically reduced.

In this paper, we propose a novel algorithm for efficient clustering. This algorithm is a modified FCM called the psFCM algorithm, which significantly reduces the computation time required to partition a dataset into desired clusters. We find the actual cluster center by using a simplified set of the original complete dataset. It refines the initial value of the FCM algorithm to speed up the convergence time. Our experiments show that the proposed psFCM algorithm is on average four times faster than the original FCM algorithm. We also demonstrate that the quality of the proposed psFCM algorithm is the same as the FCM algorithm.

1. Introduction

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. 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 many fuzzy clustering methods being introduced [1]. The fuzzy C-means (FCM) algorithm is widely used. It is based on the concept of fuzzy C-partition, which was introduced by Ruspini [2], developed by Dunn [3], and generalized by Bezdek [4,5]. The FCM algorithm and its derivatives have been used very successfully in many applications, such as pattern recognition [6], classification [7], data mining [8], and image segmentation [9,10]. It has also been used for data analysis and modeling [11,12] etc.

Normally, the FCM algorithm consists of several execution steps. In the first step, the algorithm selects C initial cluster centers from the original dataset randomly. Then, in later steps, after several iterations of the algorithm, the final result converges to the actual cluster center. Therefore, choosing a good set of initial cluster centers is very important for an FCM algorithm. However, it is difficult to select a good set of initial cluster centers randomly. If a good set of initial cluster centers is chosen, the algorithm may take less iterations to find the actual cluster centers.

To show that selecting a set of initial cluster centers that approximates the actual cluster centers can reduce the number of iterations and improve the system performance, we use Figures 1 and 2 for illustration. Using the target tracking by initializing each iteration in the procedure with the clustering result from the previous one can speed up the convergence significantly. Since the number of iterations required in the FCM algorithm strongly depends on the initial cluster centers, the goal of our proposed method is to find a good set of initial cluster centers.

In [13], Cheng et al. propose the *multistage random sampling* FCM algorithm. It is based on the assumption that a small subset of a dataset of feature vectors can be used to approximate the cluster centers of the complete dataset. Under this assumption FCM is used to compute the cluster centers of an appropriate size subset of the original dataset. After obtaining the cluster centers of this small subset, the subset of data is merged with an additional small, randomly selected subset of the remaining unprocessed feature vectors to form a larger subset that is processed by FCM. The previously calculated cluster centers are used for the initialization of

0-7695-1119-8/01 $17.00 © 2001 IEEE

the fuzzy partition matrix of this newly formed set. The procedure above is repeated until the size of the feature vectors matrix used in calculations is large enough to approximate the actual cluster center of the full dataset. The resulting cluster centers are then used for the initialization of the fuzzy partition matrix used by FCM when it is applied to the original dataset. This results in a faster convergence for the FCM algorithm.

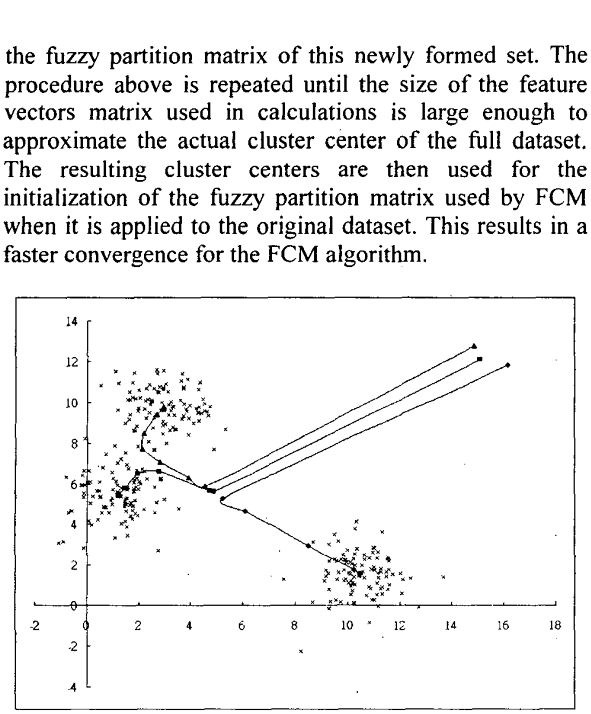

Figure 1. An example showing the convergence paths from three distant initial cluster centers to the final actual cluster centers after seven iterations by using the FCM algorithm.

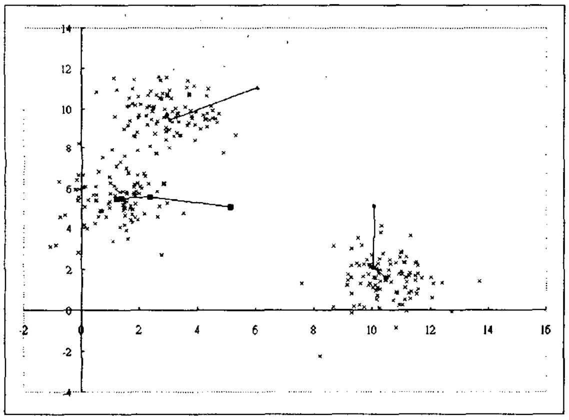

Figure 2. An example of improved initial cluster centers that are near the final cluster centers requiring only three FCM iterations.

The FCM algorithm and its derivatives have the iterative nature of an algorithm. In addition, their calculation often involves a huge number of membership matrices and candidate cluster centers matrices. It is a computationally intensive method. In our study, we mitigate the time problem by simplifying the computation and reducing the number of iterations required to converge. The idea of the proposed method is to simplify the dataset and find an initial candidate set of cluster centers as close as possible to the actual cluster centers. This will reduce the number of iterations and improve the execution performance. The initial cluster center found by the proposed algorithm approximates the actual cluster center very well.

This efficient algorithm for improving the FCM is called the *partition simplification* FCM (psFCM). It is divided into two phases. In *Phase I*, we first partition the dataset into some small block cells using the *k-d tree* method [14] and reduce the original dataset into a *simplified dataset* with unit blocks as described in our previous work [15]. 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 a small number of unit blocks' centroids, i.e., the simplified dataset. Secondly, we find the actual cluster center of this simplified dataset by the FCM algorithm. In *Phase II*, it is a standard process of the FCM with the cluster centers initialized by the final cluster centers from Phase I. The execution performance of the psFCM is much better than that of the FCM and its derivatives.

The rest of the paper is organized as follows. The review of the previously proposed approach for the FCM algorithm is in Section 2. In Section 3, we discuss the proposed algorithm. In Section 4, we show the experimental results and discuss the time complexity and accuracy issue. Finally, in Section 5, we conclude the paper.

2. Related Work

In this section, we briefly describe the Fuzzy C-means algorithm. Consider a set of unlabeled patterns $X=\{x_1,x_2,\dots,x_N\}$, $x_i \in R^f$, where N is the number of patterns and f is the dimension of *pattern vectors* (features). The FCM algorithm focuses on minimizing the value of an *objective function*. The objective function measures the quality of the partitioning that divides a dataset into C clusters.

The FCM algorithm measures the quality of the partitioning by comparing the distance from pattern x_i to the current candidate cluster center w_j with the distance from pattern x_i to other candidate cluster centers. The objective function is an optimization function that calculates the *weighted within-group sum of squared errors* (WGSS) as follows [5]:

$$\text{Minimize} \quad J_m(U,W) \equiv \sum_{j=1}^{C}\sum_{i=1}^{N}(\mu_{ij})^m d_{ij}^2 \qquad (1)$$

where:

N: the number of patterns in X
C: the number of clusters

U: the membership function matrix; the elements of U are μ_{ij}

μ_{ij}: the value of the membership function of the i^{th} pattern belonging to the j^{th} cluster

d_{ij}: the distance from x_i to w_j, viz., $d_{ij}=\left\|x_i - w_j^{(t)}\right\|$; where $w_j^{(t)}$ denotes the cluster center of the j^{th} cluster for the t^{th} iteration

W: the cluster center vector

m: the exponent on μ_{ij}; to control fuzziness or amount of clusters overlap

The FCM algorithm focuses on minimizing J_m, subject to the following constraints on U:

$$\mu_{ij} \in [0,1] \text{ , } i=1,\ldots,N \text{ , and } j=1,\ldots,C \qquad (2)$$

$$\sum_{j=1}^{C} \mu_{ij} = 1 \text{ , } i=1,\ldots,N \qquad (3)$$

$$0 < \sum_{i=1}^{N} \mu_{ij} < N \text{ , } j=1,\ldots,C \qquad (4)$$

Function (1) describes a constrained optimization problem, which can be converted to an unconstrained optimization problem by using the Lagrange multiplier technique.

$$\mu_{ij}^{(t)} = \frac{1}{\sum_{l=1}^{C}\left(\frac{d_{ij}}{d_{il}}\right)^{\frac{2}{(m-1)}}} \text{ , } i=1,\ldots,N \text{ , } j,l=1,\ldots,C \qquad (5)$$

$$\text{If } d_{ij}=0 \text{ then } \mu_{ij}=1 \text{ and } \mu_{ij}=0 \quad \text{for } l \neq j \qquad (6)$$

$$W_j^{(t)} = \frac{\sum_{i=1}^{N}(\mu_{ij}^{(t-1)})^m x_i}{\sum_{i=1}^{N}(\mu_{ij}^{(t-1)})^m} \text{ , } j=1,\ldots,C \qquad (7)$$

The FCM algorithm starts with a set of *initial cluster centers* (or *arbitrary membership values*). Then, it iterates the two updating functions (5) and (7) at the t^{th} iteration until the cluster centers are stable or the objective function in (1) converges to a local minimum. The complete algorithm consists of the following steps:

Step 1: Given a fixed number C, initialize the cluster center matrix $W^{(0)}$ by using a random generator from the original dataset. Record the cluster centers, set t=0, m = 2, and decide ε , where ε is a small positive constant.

Step 2: Initialize the membership matrix $U^{(0)}$ by using functions (5) and (6).

Step 3: Increase t by one. Compute the new cluster center matrix (candidate) $W^{(t)}$ by using (7).

Step 4: Compute the new membership matrix $U^{(t)}$ by using functions (5) and (6).

Step 5: If $\| U^{(t)} - U^{(t-1)} \| < \varepsilon$ then stop, otherwise go to step 3.

3. Our Proposed Algorithm

In this section, we propose a novel method for the clustering problem. The proposed psFCM algorithm can speed up the overall computation time and reduce the total number of calculations. The main idea of the proposed algorithm is to *refine the initial cluster centers* (initial prototypes). It finds a set of initial cluster centers that is very close to the actual cluster centers of the dataset.

First, we give an introductory explanation of the proposed algorithm followed by a formal description. The psFCM algorithm consists of two phases. Phase I is a sequence of processes that refines the initial cluster centers. In the first stage, we partition the dataset into several *unit blocks* by using the k-d tree method [14]. There must be at least one pattern in each unit block. Thus, the actual number of unit blocks depends on the size and pattern distribution of the dataset. For each unit block, we calculate the centroid of patterns in the unit block. The centroid of patterns will be used to represent all the patterns in this unit block. By doing so, a dataset X_N can be drastically reduced to a simplified dataset X_{ps} containing the centroids of the original patterns.

In the second stage, we apply the FCM algorithm to find the cluster centers of the simplified dataset $\overline{X}_{ps} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_{ps})$, $\overline{x}_i \in R^f$. The number of centroids in the simplified dataset is N_{ps}. It is equivalent to the number of unit blocks N_{ub}. Since $N_{ps} << N$, we may reduce the number of calculations of the norm distance. This reduces the overall computation time.

The cluster centers found in Phase I are used in Phase II. In Phase II, we apply the FCM algorithm to find the actual cluster center of the dataset. That is, the cluster centers found in Phase I are the initial values of the fuzzy partition matrix used by the FCM algorithm in Phase II. The process in Phase II converges quickly because the initial cluster centers are near the location of the actual cluster centers.

Before we show details of the psFCM algorithm, there are several parameters that need to be defined. They are the number of clusters C, the *weight* exponent m, the number of *unit blocks* N_{ub} (the number of splits with the k-d tree method), and *the stopping conditions* ε_1 and ε_2 in Phases I and II, respectively. Here, we let the weight exponent m be 2. The number of unit blocks N_{ub} depends on the total number of patterns N and the distribution of the dataset. The value of ε_1 and ε_2 is decided from experiments as explained in Section 4.

The proposed psFCM algorithm consists of two phases as follows:

3.1. Phase I : Refine initial prototypes for fuzzy C-means clustering

Stepl:

First, we partition the dataset into unit blocks by using the k-d tree method. The splitting priority depends on the scattered degree of data values for each dimension. If one dimension has a higher scattered degree, it has a higher priority to be split. The scattered degree is defined as the distribution range and the standard deviation of the feature. The formula of the scattered degree is as follows:

$$R_i = \frac{(X_{\max} - X_{\min})_i}{\sigma_i} \quad , \; i = 1,..f \qquad (8)$$

Where

R_i : the scattered degree for the i^{th} feature (dimension)

X_{max}: the maximum value of pattern in the i^{th} feature

X_{min} : the minimum value of pattern in the i^{th} feature

σ_i : the standard deviation of all patterns in the i^{th} feature

f: the number of features (dimensions)

The k-d tree is a kind of binary partition based on the difference between the maximum and minimum values of the partition dimension. Thus, we may easily acquire the range of each block in each dimension. If the number of splits is p, $N_{ub} \leqq 2^p$.

After splitting the dataset into unit blocks, every unit block may contain some sample patterns. There must be at least one pattern in each unit block. If there is no sample pattern in a unit block, the unit block will then be discarded. We do not consider such a unit block in the following steps. Figure 3 gives an example where a two-dimension dataset is divided into several unit blocks. Here, we only have to scan the database twice to identify the location of each sample pattern.

Step 2:

After splitting the dataset into unit blocks, we calculate the centroid $\overline{X}_i$ for each unit block that contains some sample patterns. The centroid $\overline{X}_i$ represents all sample patterns in this unit block. Then, we use all of these centroids $\overline{X}_i$ to denote the original dataset. Figure 4 gives an example in which original patterns are represented by all the computed centroids $\overline{X}_i$.

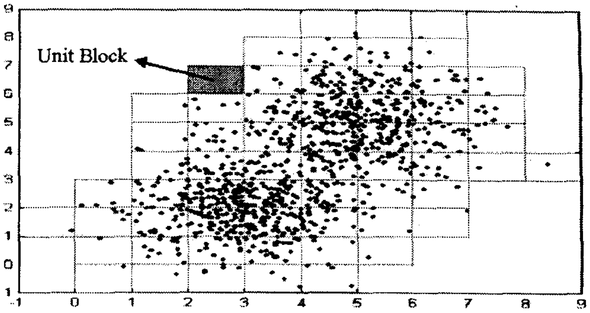

Figure 3. Partitioning the original dataset into several unit blocks.

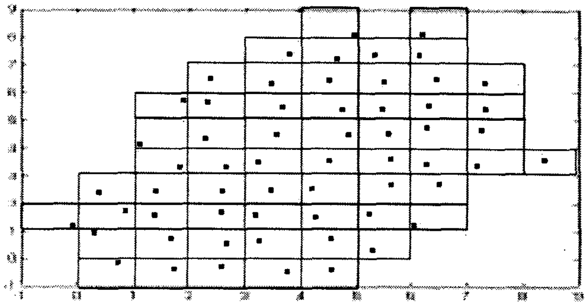

Figure 4. The simplified dataset of Figure 3.

In addition, each centroid contains statistical information of the patterns in each unit block. These include the number of patterns in a unit block (*WUB*) and the linear sum of all patterns in a unit block (*LSUB*). When we scan the database the second time, it also finds the statistics of each dimension. These statistics will be used when the algorithm calculates new candidate cluster centers, which improves the system performance.

The formula of calculating the centroid in the i^{th} unit block is as follows:

$$\overline{X}_i = \frac{LSUB_i}{WUB_i} \qquad (9)$$

The algorithm to split a dataset into unit blocks and to form the simplified dataset is shown in Figure 5.

Step 3:

Initialize the cluster center matrix $W^{(0)}$ by using a random generator from the dataset, record the cluster centers, and set t=0.

Step 4:

Initialize the membership matrix $U^{(0)}$ by using functions (5) and (6) with the simplified dataset $\overline{X}_{ps} = (\overline{x}_1, \overline{x}_2, ..., \overline{x}_{ps})$ and $i = 1,..., N_{ps}$.

```
Proc partition_dataset (original dataset)
  for each dimension $D_f$ of the dataset
      /* Find the range for dimension $D_f$ */
      Range_Max[f] = the maximum value of dimension $D_f$
      Range_Min[f] = the minimum value of dimension $D_f$
      /* Calculate the interval of segment it would partition for dimension $D_f$ */
      /* Num_of_Split[f] is the number of splits for dimension $D_f$ */
      Interval_of_Seg[f] = ( Range_Max[f] – Range_Min[f] ) / Num_of_Split[f]
  for each pattern $X_i$
      /* Calculate the UB to which pattern $X_i[f]$ belongs */
      for each dimension $D_f$ of $X_i$, named $X_i[f]$
      Point_in_Dim[f] = ( $X_i[f]$ – Range_Min[f] ) / Interval_of_Seg[f]
      /* Use the value of Point_in_Dim[f] to calculate the $UB_i$ to which $X_i[f]$ belongs, named UB_Location of $X_i[f]$ */
  /* Calculate LSUB and WUB for each $UB_i$ */
  UB_process ( $X_i[f]$, UB_Location )
  /* Compute $\bar{X}_i$ : Centroid of Unit Block i */
      $\bar{X}_i = \frac{LSUB_i}{WUB_i}$
End partition_dataset
```

Figure 5. The algorithm to partition the original dataset to form the simplified dataset.

Step 5:

Increase t (i.e., t=t+1); compute a new cluster center matrix (candidate) $W^{(t)}$ by using function (10).

$$W_j^{(t)} = \frac{\sum_{i=1}^{N_{ps}} (\mu_{ij}^{(t-1)})^m n_i \bar{x}_i}{\sum_{i=1}^{N_{ps}} (\mu_{ij}^{(t-1)})^m n_i}, \quad i = 1,\dots,N_{ps} \quad , \quad j = 1,\dots,C \qquad (10)$$

where n_i denotes the number of patterns of the i^{th} unit block.

Step 6:

Compute the new membership matrix $U_{N_{ps}}^{(t)}$ by using functions (5) and (6) with the simplified dataset $\bar{X}_{ps} = (\bar{x}_1, \bar{x}_2, \dots, \bar{x}_{ps})$ and $i = 1,\dots,N_{ps}$.

Step 7:

If $\left\| U_{N_{ps}}^{(t)} - U_{N_{ps}}^{(t-1)} \right\| \geqq \varepsilon_1$, go to Step 5, otherwise go to Phase II.

3.2. Phase II: Find the actual cluster centers for the dataset

Step1:

Initialize the fuzzy partition matrix $U^{(0)}$ by using the results of $W^{(t)}$ from Phase I with functions (5) and (6) for the dataset X.

Step 2:

Follow Step 3 to Step 5 of the FCM algorithm discussed in Section 2 using the stopping condition ε_2.

4. Experiments and Results

In this section, we discuss our experimental results, the time complexity of the proposed method, and its accuracy. The datasets used in the experiments are described in Section 4.1. Section 4.2 shows the experimental results. In Section 4.3, we discuss our findings.

4.1 Dataset Description

In this paper, we focus on improving the time complexity of the fuzzy C-means algorithm. To show that the result of our proposed algorithm is correct and more efficient than the other algorithms, we performed a series of experiments. Table 1 shows the datasets used in the experiments, where N is the number of patterns, C is the number of clusters, and f is the dimension of pattern vectors (features).

The datasets are generated by using the following criteria:

1) The datasets from D_1 to D_{10} are generated from a *normal distribution.* Here, we generate normally distributed data points (patterns) by using the *Marsaglia's Polar method* [17]. The *standard deviation* of these data points in each cluster is 1. We randomly select C data points from the range [0,4C]. These C data points may be treated as the mean value of the normal distribution of each cluster. Every cluster has the same number of data points N/C that is around a specific mean value, and *clusters can overlap.*
2) The datasets from R_1 to R_4 are generated from a *uniform distribution. Any two clusters cannot overlap.* The number of data points (patterns) in each cluster is the same, which is N/C.
3) The datasets from R_5 to R_{10} are also generated from a *uniform distribution.* However, *clusters can overlap.* The number of data points in each cluster is different. Each cluster has $N_i = i \times \frac{2N}{C(C+1)}$ data points, where i = 1,2,…,C.

Table 1. Description of the experiment datasets.

Dataset	Size (N)	Dimensionality (f)	No. of Clusters (C)	Characteristic (distribution)	Range
D_1	32k	2	8	Normal	[0,4C]
D_2	32k	2	16	Normal	[0,4C]
D_3	32k	2	32	Normal	[0,4C]
D_4	64K	2	8	Normal	[0,4C]
D_5	64k	2	16	Normal	[0,4C]
D_6	64k	2	32	Normal	[0,4C]
D_7	64k	2	64	Normal	[0,4C]
D_8	128k	2	8	Normal	[0,4C]
D_9	128k	2	16	Normal	[0,4C]
D_{10}	128k	2	32	Normal	[0,4C]
R_1	16k	2	4	Random	[0,1]
R_2	16k	2	8	Random	[0,1]
R_3	32k	2	4	Random	[0,1]
R_4	32k	2	8	Random	[0,1]
R_5	16k	2	4	Random	[0,1]
R_6	16k	2	8	Random	[0,1]
R_7	16k	2	16	Random	[0,1]
R_8	32k	2	4	Random	[0,1]
R_9	32k	2	8	Random	[0,1]
R_{10}	32k	2	16	Random	[0,1]

4.2 Experimental Results

The experiment datasets (D_1 ~ D_{10} and R_1 ~ R_{10}) are used to run on two personal computers. One is a *Pentium III* personal computer with the following specification: a clock rate of 800MHz and memory size of 512 Mbytes. The other is a *Celeron* personal computer with a clock rate of 433Hz and memory size of 64 Mbytes.

When we run the experiments using the FCM algorithm, different random initialization sets may result in different numbers of convergence iterations. Therefore, we run the proposed psFCM algorithm and the FCM algorithm with each dataset in ten trials. In each trial, the initial cluster centers are initialized in a random fashion. The same initial cluster centers are used by both the psFCM algorithm and the FCM algorithm.

The execution performance of the psFCM algorithm and the FCM algorithm is shown in Tables 2 and 3. Here, the comparison is based on the two factors: the factor reduction in overall time (FRT) and the factor reduction in distance calculations (FRD) [16]. From the results in Tables 2 and 3, the execution performance of the psFCM algorithm is approximately four times better than that of the FCM algorithm.

In the experiment of the psFCM algorithm, the stopping condition ε_1 of Phase I is set to 0.1. The number of convergence iterations may be reduced in Phase II if we let ε_1 have a smaller value. The stopping condition ε_2 for Phase II in the psFCM algorithm and the FCM algorithm is 0.4 for both.

Table 2. The overall result of the normal distribution datasets D_1 to D_{10} where the number of splits for the k-d tree in the psFCM is 14.

Datasets	No. of Clusters	FCM		psFCM			
		Total time (Seconds)	Distance calculation	Total time (Seconds)	Distance calculation	FRT	FRD
D_1	8	295	5068800	55	1026346	5.4	4.94
D_2	16	680	10854400	167	2826976	4.07	3.84
D_3	32	1207	17408000	353	5368832	3.42	3.24
D_4	8	1091	7270400	210	2578333	5.2	4.61
D_5	16	2849	17817600	511	3560832	5.58	5.0
D_6	32	6586	38092800	2139	12958925	3.08	2.94
D_7	64	15033	97484800	5695	38018150	2.64	2.56
D_8	8	2283	14950400	348	2658806	6.56	5.62
D_9	16	5423	33587200	1153	7894640	4.7	4.2
D_{10}	32	12051	87654400	3038	23794278	3.97	3.68

Table 3. The overall result of the uniform distribution datasets R_1 to R_{10} where the number of splits for the k-d tree in the psFCM is 13.

Datasets	No. of Clusters	FCM		psFCM			
		Total time (Seconds)	Distance calculation	Total time (Seconds)	Distance calculation	FRT	FRD
R_1	4	59	614400	19	205760	3.11	2.99
R_2	8	186	1843200	63	655590	2.95	2.81
R_3	4	129	1331200	29	335872	4.45	3.96
R_4	8	332	3276800	65	720858	5.11	4.55
R_5	4	86	870400	23	240277	3.74	3.62
R_6	8	418	4069382	86	869598	4.86	4.68
R_7	16	1192	10848973	339	3106970	3.52	3.49
R_8	4	126	1305600	30	343549	4.2	3.8
R_9	8	631	6143232	86	929968	7.34	6.61
R_{10}	16	2277	20679629	395	3727757	5.76	5.55

To get the simplified dataset, the original dataset is divided into many equal size unit blocks by the psFCM algorithm. As mentioned in Section 3, all the patterns in one unit block are replaced by the centroid of the unit block. From the results of the experiments, we show that the proposed algorithm reduces a significant amount of time in Phase I.

To understand the relationship between the number of splits with the k-d tree method and the execution performance (speedup), we select some datasets (with different *N* or *C*) and compare the execution performance of these datasets by splitting them into a different number of unit blocks. Here, we measure the execution performance by using FRT. The results are shown in Figures 6 and 7. Figure 6 depicts the relationship between the number of splits and the speedup of execution time for datasets *R2* and *R4* whose sizes are 16k and 32k, respectively, while the number of clusters is the same (8 clusters). Figure 7 shows the relationship between the number of splits and the speedup of execution time for datasets *R1* and *R2* whose sizes are the same (16k) while the numbers of clusters are 4 and 8 clusters, respectively. We find the optimal number of splits is around 12. We also investigate the relationship between the number of clusters and the speedup of execution time. The experiment result is shown in Figure 8. The normal distribution dataset has a size of 64K, and the number of splits is 14. The decline of the speedup at 32 and 64 clusters can be rectified by increasing the number of splits. For example, when we adjust the number of splits to 16, the performance for the 64 clusters can be improved to a speedup of 3.81.

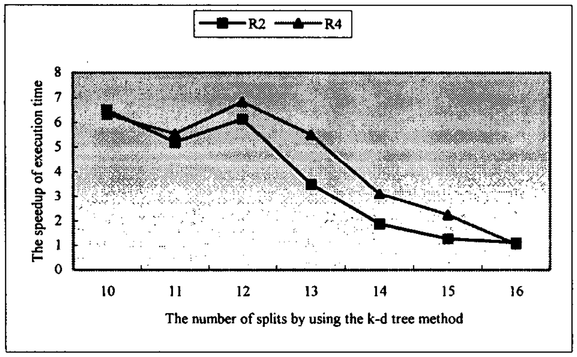

Figure 6. The relationship between the number of splits and the speedup of execution time for datasets *R2* and *R4*.

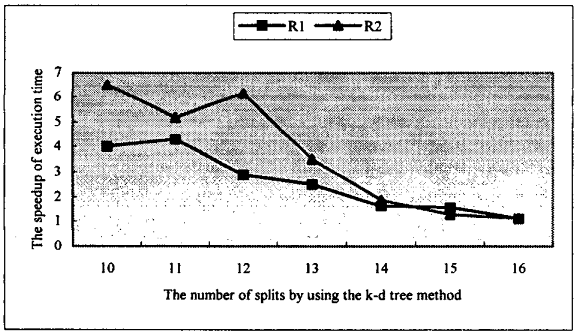

Figure 7. The relationship between the number of splits and the speedup of execution time for datasets *R1* and *R2*.

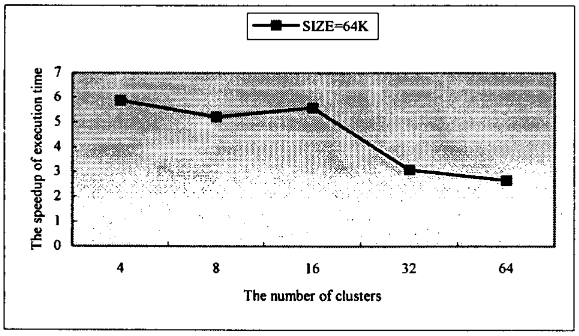

Figure 8. The relationship between the number of clusters and the speedup of execution time.

4.3 Discussions

From the experimental results described in the last section, the proposed psFCM algorithm has approximately the same speedup for the patterns of normal distribution as well as uniform distribution. In general, our method works well for most kinds of datasets.

In Phase I of the psFCM algorithm, the cluster centers found by using the simplified dataset is very close to the actual cluster centers. Phase II converges quickly if we use these cluster centers from Phase I as the initial cluster centers of Phase II. From the experiments, in most cases, Phase II converges in only a few iterations. For the psFCM algorithm, the system requires more iterations to converge if the stopping condition ε_1 is smaller. However, the number of patterns used in Phase I of the proposed algorithm is N_{ps}. The total number of norm distance calculations by the proposed algorithm is much smaller than the FCM algorithm because $N_{ps} << N$.

For the FCM algorithm, the system has to calculate the norm distance from each pattern to every candidate cluster center in each iteration. After calculating the norm distance, the system computes the membership matrix. Therefore, if the dimension of a dataset is f, the time complexity to calculate the distance is $O(f)$. If there are N patterns and C clusters, the time complexity to calculate the membership matrix for each iteration is $O(fNC)$. Thus, the time complexity is proportional to the number of patterns N and the number of clusters C. However, we have demonstrated that the time complexity to calculate the membership matrix for each iteration using the psFCM algorithm is $O(fN_{ps}C)$.

Here, we randomly select the initial cluster centers in Step 3 of the psFCM algorithm. Generally, the density of patterns near the cluster center is high. As mentioned in Section 3, the psFCM algorithm divides the dataset into unit blocks. The number of patterns in each unit block may be different. That is, the density of patterns in each unit block is different. We may randomly select C initial cluster centers $W_j(0)$ from the unit blocks with a higher pattern density. This will provide a better result.

The FCM algorithm does not guarantee that the cost function will converge to the local minimum, but it may converge to some saddle point.

5. Conclusions

In this paper, we proposed a novel method for efficient clustering that is better than the FCM algorithm. We reduce the computation cost and improve the performance by finding a good set of initial cluster centers.

The psFCM algorithm divides a dataset into several unit blocks. The centriods of unit blocks replace the patterns and form a new dataset, the simplified dataset. As mentioned in Section 4, the simplified dataset decreases the time complexity of computing the membership matrix from $O(fNC)$ to $O(fN_{ps}C)$ in every iteration. The number of iterations needed to converge in Phase II of the psFCM algorithm is also less than the number of convergence iterations of the FCM algorithm. From the experimental results, we have shown that the proposed algorithm improves the speedup in the execution time and the distance calculation. For large datasets, the psFCM algorithm improves the performance even more. We also demonstrate that the quality of our algorithm is the same as the FCM algorithm.

For a fair comparison, the initialization in Phase I of the psFCM algorithm and the FCM algorithm is determined randomly. We have also found in the psFCM algorithm that an initial cluster center selected from a unit block with a higher density is closer to the actual cluster center. This is a feature that cannot be found using the FCM algorithm and its derivatives. In future work, we will study this feature more thoroughly.

Reference

[1] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler, "*Fuzzy cluster analysis*," Wiley Press, New York, 1999.

[2] E. Ruspini, "*Numerical methods for fuzzy clustering*," Information Sciences, Vol. 2, 1970, pp. 319-350.

[3] J.C. Dunn, "*A fuzzy relative of the ISODATA process and its use in detecting compact, well separated clusters*," Cybernetics, Vol. 3, 1974, pp. 95-104.

[4] J.C. Bezdek, "*Cluster validity with fuzzy sets*," Cybernetics, Vol. 3, 1974, pp. 58-73.

[5] J.C. Bezdek, "*Pattern recognition with fuzzy objective function algorithms*," Plenum Press, New York, 1981.

[6] K.H. Chuang, M.J. Chiu, C.C. Lin, and J.H. Chen, "*Model-free functional MRI analysis using Kohonen clustering neural network and fuzzy C-means*," IEEE Trans. On Medical Imaging, Vol. 18 (12), 1999, pp. 1117-1128.

[7] N.S. Iyer, A. Kandel, and M. Schneider, "*Feature-based fuzzy classification for interpretation of mammograms*," Fuzzy Sets and Systems, Vol. 114, 2000, pp. 271-280.

[8] K. Hirota and W. Pedrycz, "*Fuzzy computing for data mining*," Proceedings of the IEEE, Vol. 87(9), 1999, pp. 1575-1600.

[9] W.E. Phillips, R.P. Velthuizen, S. Phuphanich, L.O. Hall, L.P. Clark, and M.L. Silbiger, "*Application of fuzzy c-means segmentation technique for tissue differentiation in MR images of a hemorrhagic glioblastoma multifrome*," Magnetic Resonance Imaging, Vol. 13(2), 1995, pp. 277-290.

[10] M.R. Rezaee, P.M.J. Zwet, B.P.E. Lelieveldt, R.J. Geest, and J.H.C. Reiber, "*A multiresolution image segmentation technique based on pyramidal segmentation and fuzzy clustering*," IEEE Trans. On Image Processing, Vol. 97, 2000, pp. 1238-1248.

[11] P. Teppola, S.P. Mujunen, and P. Minkkinen, "*Adaptive fuzzy c-means clustering in process monitoring*," Chemometrics and Intelligent Laboratory System, Vol. 45, 1999, pp. 22-38.

[12] X. Chang, W. Li, and J. Farrell, "*A C-means clustering based fuzzy modeling method*," The Ninth IEEE International Conference on Fuzzy Systems, Vol. 2, 2000, pp. 937-940.

[13] T.W. Cheng, D.B. Goldgof, and L.O. Hall, "*Fast fuzzy clustering*," Fuzzy Sets and Systems, Vol. 93, 1998, pp. 49-56.

[14] J.L. Bentley, "*Multidimensional binary search trees used for associative searching*," Communications of the ACM, Vol. 18(9), 1975, pp. 509-517.

[15] D.L. Yang, J.H. Chang, M.C. Hung, and J.S. Liu, "*An efficient K-means-based clustering algorithm*," Proceedings of The First Asia-Pacific Conference on Intelligent Agent Technology, Dec. 1999, pp. 269-273.

[16] K. Alsabti, S. Ranka, and V. Singh, "*An Efficient K-Means Clustering Algorithm*," PPS/SPDP Workshop on High Performance Data Mining, 1997.

[17] G. Marsaglia, "*Random variables and computers*," Information Theory Statistical Decision Functions Random Process, 1962, pp. 499-510.