Big Data and Information Analytics
©American Institute of Mathematical Sciences
Volume **1**, Number **2&3**, April & July **2016**
doi:10.3934/bdia.2016007
pp. **247–259**

SCALABLE CLUSTERING BY TRUNCATED FUZZY c-MEANS

Guojun Gan

Department of Mathematics
University of Connecticut
341 Mansfield Road, Storrs, CT 06269-1009, USA

Qiujun Lan

Business School
Hunan University
Changsha, Hunan 410082, China

Shiyang Sima

Columbian College of Arts & Sciences
George Washington University
Washington, D.C., 20052, USA

(Communicated by Weidong Bao)

Abstract. Most existing clustering algorithms are slow for dividing a large dataset into a large number of clusters. In this paper, we propose a truncated FCM algorithm to address this problem. The main idea behind our proposed algorithm is to keep only a small number of cluster centers during the iterative process of the FCM algorithm. Our numerical experiments on both synthetic and real datasets show that the proposed algorithm is much faster than the original FCM algorithm and the accuracy is comparable to that of the original FCM algorithm.

1. **Introduction.** Data clustering refers to a process of dividing a set of items into homogeneous groups or clusters such that items in the same cluster are similar to each other and items from different clusters are distinct [10, 1]. As one of the most popular tools for data exploration, data clustering has found applications in many scientific areas such as bioinformatics [21, 26], actuarial science and insurance [11, 13], image segmentation [20, 25], to name just a few.

During the past six decades, many clustering algorithms have been developed by researchers from different areas. These clustering algorithms can be divided into two groups: hard clustering algorithms and fuzzy clustering algorithms. In hard clustering algorithms, each item is assigned to one and only one cluster; In fuzzy clustering algorithms, each item can be assigned to one or more clusters with some degrees of membership. Examples of hard clustering algorithms include the k-means algorithm [27], which is one of the most widely used clustering algorithm. The FCM (Fuzzy c-means) algorithm [9, 4, 3] is a popular fuzzy clustering algorithm.

2010 *Mathematics Subject Classification.* Primary: 62H30, 68T10, 91C20; Secondary: 62P10.
Key words and phrases. Data clustering, fuzzy c-means, scalable clustering.

The FCM algorithm is formulated to minimize an objective function. Let $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a dataset containing n points. Let k be the desired number of clusters. Then the objective function of the FCM algorithm is defined as

$$Q(U, Z) = \sum_{l=1}^{k} \sum_{i=1}^{n} u_{il}^{\alpha} \|\mathbf{x}_i - \mathbf{z}_l\|^2, \tag{1}$$

where $U = (u_{il})_{n \times k}$ is an $n \times k$ fuzzy k partition matrix, $\alpha > 1$ is the fuzzifier, $Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}$ is a set of k centers, and $\|\cdot\|$ is the L^2-norm or Euclidean distance. Here a fuzzy k partition of a dataset of n points is an $n \times k$ matrix that satisfies the following conditions:

$$u_{il} \in [0, 1], \quad i = 1, 2, \ldots, n, \; l = 1, 2, \ldots, k, \tag{2a}$$

$$\sum_{l=1}^{k} u_{il} = 1, \quad i = 1, 2, \ldots, n, \tag{2b}$$

$$\sum_{i=1}^{n} u_{il} > 0, \quad l = 1, 2, \ldots, k. \tag{2c}$$

Similar to the k-means algorithm, the FCM algorithm employs an iterative process to minimize the objective function.

The FCM algorithm has some advantages over the k-means algorithm. For example, the FCM algorithm can reduce the number of local minima of the objective function [22]. However, the FCM algorithm is not efficient for dividing a large dataset into many clusters. Examples of such situations include clustering millions of web pages into a thousand categories [5] and clustering hundreds of thousands insurance policies into a thousand clusters in order to select a thousand representative policies [11, 13]. This inefficiency is caused by the following two factors. First, the FCM algorithm needs to store the full fuzzy partition matrix, which contains nk elements. Second, the FCM algorithm needs to calculate nk distances at each iteration.

In this paper, we propose a modified version of the FCM algorithm, called the TFCM (Truncated FCM) algorithm, to address the aforementioned drawback of the FCM algorithm. In the TFCM algorithm, a subset of the full fuzzy partition matrix is stored and the number of distance calculations at each iteration is reduced. The idea of the TFCM algorithm stems from the insight that when k is large, a data point belongs to only a few clusters with high degrees of membership. As a result, we can ignore the clusters with low degrees of membership while preserving the overall quality of the clustering.

The remaining part of this paper is organized as follows. In Section 2, we give a brief review of relevant work. In Section 3, we introduce the TFCM algorithm in detail. In Section 4, we demonstrate the performance of the TFCM algorithm using numerical experiments. Finally, we conclude the paper with some remarks in Section 5.

2. **Related work.**

As one of the most popular fuzzy clustering algorithms, the FCM algorithm was originally proposed by [9] and later modified by [4]. Many improvements of the FCM algorithm have been proposed since its introduction. In this section, we give a brief review of research work related to the efficiency of the FCM algorithm.

[6] proposed the AFCM (Approximate FCM) algorithm by replacing some variates in the FCM equations with integer-valued or real-valued estimates. The AFCM algorithm was developed to process digital images interactively. In the implementation of the AFCM algorithm, the fuzzy memberships u_{il} are approximated by real numbers with three decimal places and stored as integers in $[0, 1000]$ in memory. In addition, the AFCM algorithm stores six internal tables in memory and uses a table lookup approach to eliminate the use of exponentiation operators in the updating of the cluster centers and fuzzy memberships. Experimental results show that the runtime of each iteration of the AFCM algorithm is reduced approximately to one sixth of that of a literal implementation of the FCM algorithm.

[7] proposed a multistage random sampling FCM algorithm, called the mrFCM algorithm, to reduce the runtime of the FCM algorithm. The mrFCM algorithm consists of two phases. In the first phase, the FCM algorithm is applied to a series of subsamples selected randomly from the whole dataset in order to find good initial cluster centers. In the second phase, the standard FCM algorithm with the initial cluster centers obtained from the first phase is applied to partition the whole dataset.

[19] proposed the psFCM (partition simplification FCM) algorithm to speed up the FCM algorithm by simplifying the computation at each iteration and reducing the number of iterations. Similar to the mrFCM algorithm [7], the psFCM algorithm also consists of two phases. In the first phase, the kd-tree method is first used to partition the whole dataset into small blocks. All points in a block are represented by the centroid of the block. In this way, a large dataset is reduced to a simplified dataset that is much smaller than the original dataset. Then the FCM algorithm is applied to the simplified dataset to obtain the actual cluster centers. In the second phase, the FCM algorithm with the cluster centers obtained from the first phase is applied to partition the original dataset.

[23] proposed a modified version of the FCM algorithm by eliminating the need to store the fuzzy partition matrix U. In the modified version, updating the cluster centers and updating the fuzzy memberships are combined into a single step. The original FCM algorithm has a time complexity of $O(nk^2d)$, but this modified version reduces the time complexity to $O(nkd)$, where n, k, and d denote the number of data points, the desired number of clusters, and the number of attributes, respectively.

[24] proposed the PFCM (Parallel FCM) algorithm for clustering large datasets by using the Message Passing Interface (MPI). In the PFCM algorithm with P processors, a dataset of n points is divided into P blocks of equal size so that each processor processes n/P data points. The fuzzy partition matrix is also divided into P sections so that the memberships of local data points of a processor are stored in the processor's memory. An MPI call is used to pass messages if a computation requires data points stored in other processes. The PFCM algorithm is an example of speeding up the FCM algorithm through hardware. Another example of speeding up the FCM algorithm through hardware is to use the graphics-processing unit (GPU) [28].

[17] proposed the geFFCM (generalized extensible fast fuzzy c-means) algorithm to cluster very large datasets. The geFFCM algorithm is similar to the mrFCM algorithm [7] and the psFCM algorithm [19] in the sense that divide-and-conquer strategy is used by all three algorithms. In the geFFCM algorithm, a subsample X_{SS} is drawn from the original dataset X without replacement such that the number of features for which X_{SS} and X agree is not less than a specified number. Then the

standard FCM algorithm is applied to X_{SS} to obtain the cluster centers. Finally, the cluster centers are used to obtain the fuzzy partition matrix of the original dataset.

[18] compared three different implementation of the FCM algorithm for clustering very large datasets. In particular, [18] compared the random sample and extension FCM, single-pass FCM, and on-line FCM. In addition, kernelized versions of the three algorithms were also compared. [29] proposed the FCM++ algorithm to improve the speed of the FCM algorithm by using the seeding mechanism of the K-means++ [2].

Almost all of the aforementioned algorithms aim at speeding up the FCM algorithm for large datasets. These algorithms do not scale well when the desired number of clusters is large. The algorithm proposed by [23] reduces the time complexity of the FCM algorithm from $O(nk^2d)$ to $O(nkd)$. However, the algorithm still needs to calculate nk distances at each iteration. In the next section, we propose the truncated fuzzy c-means algorithm to approximate the FCM algorithm when the desired number of clusters is large.

3. The TFCM algorithm.

In this section, we introduce the TFCM (Truncated Fuzzy c-means) algorithm.

Let $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a dataset containing n points. Let k be the desired number of clusters. A fuzzy partition matrix $U = (u_{il})_{n\times k}$ of dividing X into k clusters is an $n \times k$ matrix that satisfies the following conditions

$$u_{il} \in [0, 1], \quad i = 1, 2, \ldots, n,\ l = 1, 2, \ldots, k,$$

and

$$\sum_{l=1}^{k} u_{il} = 1, \quad i = 1, 2, \ldots, n.$$

Let T be an integer such that $1 \le T \le k$. Let $\mathcal{U}_T$ be the set of fuzzy partition matrices U such that each row of U has at most T nonzero entries. In other words, $U \in \mathcal{U}_T$ if U is a fuzzy partition matrix such that for each $i = 1, 2, \ldots, n$,

$$|\{l : u_{il} > 0\}| \le T, \tag{3}$$

where $|\cdot|$ denote the number of elements in a set.

Then the objective function of the TFCM algorithm is defined as

$$P(U, Z) = \sum_{i=1}^{n}\sum_{l=1}^{k} u_{il}^{\alpha}\left(\|\mathbf{x}_i - \mathbf{z}_l\|^2 + \epsilon\right), \tag{4}$$

where $\alpha > 1$ is the fuzzifier, $U \in \mathcal{U}_T$, $Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}$ is a set of cluster centers, $\|\cdot\|$ is the L^2-norm or Euclidean distance, and ϵ is a small positive number used to prevent division by zero. Let $I_i = \{l : u_{il} > 0\}$ for $i = 1, 2, \ldots, n$. Then we can rewrite the objective function (4) as

$$P(U, Z) = \sum_{i=1}^{n}\sum_{l\in I_i} u_{il}^{\alpha}\left(\|\mathbf{x}_i - \mathbf{z}_l\|^2 + \epsilon\right). \tag{5}$$

From Equation (5) we see that the main difference between the TFCM algorithm and the fuzzy c-means algorithm is the constraint given in Equation (3). If $T = 1$, then the TFCM algorithm becomes the k-means algorithm. If $T = k$, then the TFCM algorithm becomes the fuzzy c-means algorithm.

Theorem 3.1. *Given a set of centers Z. The fuzzy partition matrix $U \in \mathcal{U}_T$ that minimizes the objective function (4) is given by*

$$u_{il} = \frac{\left(\|\mathbf{x}_i - \mathbf{z}_l\|^2 + \epsilon\right)^{-\frac{1}{\alpha-1}}}{\sum_{s\in I_i}\left(\|\mathbf{x}_i - \mathbf{z}_s\|^2 + \epsilon\right)^{-\frac{1}{\alpha-1}}}, \quad 1 \le i \le n,\ l \in I_i, \tag{6}$$

where I_i is the set of indices of the T centers that are closest to $\mathbf{x}_i$, i.e.,

$$I_i = \{l_1, l_2, \ldots, l_T\} \tag{7}$$

with $(l_1, l_2, \ldots, l_k)$ being a permutation of $(1, 2, \ldots, k)$ such that

$$\|\mathbf{x}_i - \mathbf{z}_{l_1}\| \le \|\mathbf{x}_i - \mathbf{z}_{l_2}\| \le \cdots \le \|\mathbf{x}_i - \mathbf{z}_{l_k}\|.$$

Proof. For each $i = 1, 2, \ldots, n$, let I_i be the set of indices defined in Equation (7). We first show that for these I_i, the optimal weights are given in Equation (6). Since the rows of a fuzzy partition matrix are independent of each other, the objective function (4) is minimized if for each $i = 1, 2, \ldots, n$, the following function

$$P_i(\mathbf{u}_i, I_i) = \sum_{l\in I_i} u_{il}^{\alpha}\left(\|\mathbf{x}_i - \mathbf{z}_l\|^2 + \epsilon\right) \tag{8}$$

is minimized subject to $\sum_{l\in I_i} u_{il} = 1$, where $\mathbf{u}_i = (u_{i1}, u_{i2}, \ldots, u_{ik})$. Using the method of Lagrange multipliers, we can obtain the optimal weights by minimizing the following function

$$P_i(\mathbf{u}_i, \lambda, I_i) = \sum_{l\in I_i} u_{il}^{\alpha}\left(\|\mathbf{x}_i - \mathbf{z}_l\|^2 + \epsilon\right) + \lambda\left(\sum_{l\in I_i} u_{il} - 1\right).$$

We can obtain the optimal weights given in Equation (6) by solving the equations obtained by taking derivatives of $P_i(\mathbf{u}_i, \lambda, I_i)$ with respect to λ and u_{il} for $l \in I_i$ and equating the derivatives to zero.

Now we show that

$$P_i(\mathbf{u}_i^*, I_i) \le P_i(\mathbf{v}_i^*, J_i), \tag{9}$$

where J_i is an arbitrary subset of $\{1, 2, \ldots, k\}$ such that $|J_i| \le T$, and $\mathbf{u}_i^*$ and $\mathbf{v}_i^*$ are the optimal weights obtained from Equation (6) when the underlying index sets are I_i and J_i, respectively.

Since $\mathbf{u}_i^*$ is the vector of optimal weights, we have

$$\begin{aligned}
P_i(\mathbf{u}_i^*, I_i) &= \sum_{l\in I_i}(u_{il}^*)^{\alpha}\left(\|\mathbf{x}_i - \mathbf{z}_l\|^2 + \epsilon\right) \\
&= \sum_{l\in I_i} \frac{\left(\|\mathbf{x}_i - \mathbf{z}_l\|^2 + \epsilon\right)^{-\frac{\alpha}{\alpha-1}}}{\left(\sum_{s\in I_i}\left(\|\mathbf{x}_i - \mathbf{z}_s\|^2 + \epsilon\right)^{-\frac{1}{\alpha-1}}\right)^{\alpha}}\left(\|\mathbf{x}_i - \mathbf{z}_l\|^2 + \epsilon\right) \\
&= \sum_{l\in I_i} \frac{\left(\|\mathbf{x}_i - \mathbf{z}_l\|^2 + \epsilon\right)^{-\frac{1}{\alpha-1}}}{\left(\sum_{s\in I_i}\left(\|\mathbf{x}_i - \mathbf{z}_s\|^2 + \epsilon\right)^{-\frac{1}{\alpha-1}}\right)^{\alpha}} \\
&= \frac{1}{\left(\sum_{s\in I_i}\left(\|\mathbf{x}_i - \mathbf{z}_s\|^2 + \epsilon\right)^{-\frac{1}{\alpha-1}}\right)^{\alpha-1}}.
\end{aligned}$$

Similarly, we have

$$P_i(\mathbf{v}_i^*, J_i) = \frac{1}{\left(\sum_{s\in J_i}\left(\|\mathbf{x}_i - \mathbf{z}_s\|^2 + \epsilon\right)^{-\frac{1}{\alpha-1}}\right)^{\alpha-1}}.$$

Since $\alpha > 1$, I_i contains the indcies of the T centers that are closest to $\mathbf{x}_i$, and $|J_i| \leq T$, we have

$$\sum_{s \in I_i} \left(\|\mathbf{x}_i - \mathbf{z}_s\|^2 + \epsilon \right)^{-\frac{1}{\alpha-1}} \geq \sum_{s \in J_i} \left(\|\mathbf{x}_i - \mathbf{z}_s\|^2 + \epsilon \right)^{-\frac{1}{\alpha-1}},$$

which shows that the inequality given in Equation (9) is true. This completes the proof. □

Theorem 3.2. *Given a fuzzy partition matrix $U \in \mathcal{U}_T$. The set of centers Z that minimizes the objective function* (4) *is given by*

$$z_{lj} = \frac{\sum_{i=1}^{n} u_{il}^{\alpha} x_{ij}}{\sum_{i=1}^{n} u_{il}^{\alpha}} = \frac{\sum_{i \in C_l} u_{il}^{\alpha} x_{ij}}{\sum_{i \in C_l} u_{il}^{\alpha}}, \tag{10}$$

for $l = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, d$, where d is the number of features, z_{lj} is the jth component of $\mathbf{z}_l$, and $C_l = \{i : u_{il} > 0\}$.

The proof of Theorem 3.2 is omitted as it is similar to the result of the FCM algorithm.

Algorithm 1: Pseudo-code of the TFCM Algorithm.

Input: $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, k, T, δ, N_{max}, α
Output: U, Z

Initialize the set of cluster centers $Z^{(0)}$ by selecting k data points from X randomly;
for $i = 1$ *to* n **do**
  Calculate the distance between $\mathbf{x}_i$ and all k centers;
  Let I_i be the subset of $\{1, 2, \ldots, k\}$ such that the corresponding T centers are closest to $\mathbf{x}_i$;
  Update the weights $u_{il}^{(0)}$ for $l \in I_i$ according to Equation (6);
end
$s \leftarrow 0$;
$P^{(0)} \leftarrow 0$;
while *True* **do**
  Update the set of cluster centers $Z^{(s+1)}$ according to Equation (10);
  for $i = 1$ *to* n **do**
    Select T centers with indices in $\{1, 2, \ldots, k\}/I_i$ randomly;
    Calculate the distance between $\mathbf{x}_i$ and centers with indices in $I_i \cup J_i$;
    Update I_i with the indices of the T centers that are closest to $\mathbf{x}_i$;
    Update the weights $u_{il}^{(s+1)}$ for $l \in I_i$ according to Equation (6);
  end
  $P^{(s+1)} \leftarrow P\left(U^{(s+1)}, Z^{(s+1)}\right)$;
  if $\left|P^{(s+1)} - P^{(s)}\right| < \delta$ *or* $s \geq N_{max}$ **then**
    Break;
  end
  $s \leftarrow s + 1$;
end

The pseudo-code of the TFCM algorithm is given in Algorithm 1. The TFCM algorithm consists of two phases: the initialization phase and the iteration phase. In the initialization phase, we initialize the cluster centers to be k data points randomly selected from the dataset. We also calculate the distances between all data points and all initial centers in order to calculate the weights U. In the iteration phase, for each data point $\mathbf{x}_i$, we only calculate the distances between the data point and $2 * T$ cluster centers, which include the T existing centers saved in I_i and T centers randomly selected from the remaining $k - T$ centers. Among the $2 * T$ centers, we only keep the T centers that are closest to the point $\mathbf{x}_i$ and save them for the next iteration. The advantage of selecting some centers from the remaining centers is that it helps us to find the true center for a data point if the true center was not selected in the initialization phase or previous iterations.

Regarding how to choose a value for the parameter T, a good starting point is $d + 1$, where d is the dimension of the underlying dataset. The reason to use $T = d + 1$ is that $d + 1$ is the number of vertices of a d-dimensional simplex. In a d-dimensional dataset, a point can be surrounded by $d + 1$ sphere-shaped clusters. For example, in an one-dimensional dataset, a data point has two closest clusters. In a two-dimensional dataset, a data point has three closest clusters.

Parameter	**Default Value**
ϵ	10^{-6}
T	$d + 1$
δ	10^{-6}
N_{max}	1000
α	2

TABLE 1. Default values for some parameters required by the TFCM algorithm.

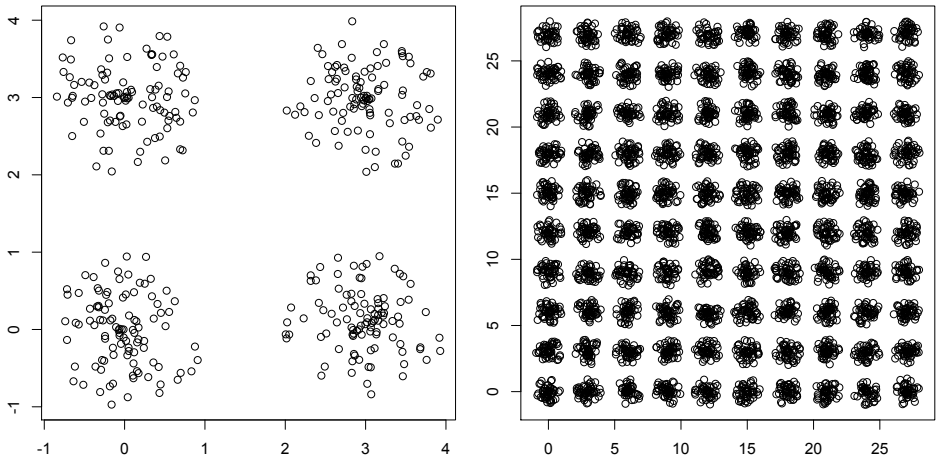
A list of default values for the parameters required by the TFCM algorithm is given in Table 1. The parameters N_{max} and δ are used to terminate the algorithm. The FCM algorithm usually converges in a few iterations. Since the TFCM algorithm only calculates the distances between data points and a small subset of the centers, it may need more iterations to converge. Hence we suggest setting the default value of the maximum number of iterations to 1000. The parameter α is the fuzzifier, which should be larger than 1.

4. **Experimental evaluation.** In this section, we present some numerical results to demonstrate the performance of the TFCM algorithm in terms of speed and accuracy. We also compare the performance of the TFCM algorithm to that of the FCM algorithm. We implemented both the TFCM algorithm and the FCM algorithm in Java. In order to make relatively fair comparison between the TFCM algorithm and the FCM algorithm, we used the same sets of initial cluster centers and the same criteria to terminate the algorithms.

4.1. **Results on synthetic data.** To show that the TFCM algorithm works, we created two synthetic datasets, which are summarized in Table 2. Both synthetic datasets are two-dimensional datasets. One dataset contains four clusters and the other dataset contains 100 clusters. Figure 1 shows the two datasets.

Dataset	Size	Dimension	Clusters
S1	400	2	4 clusters, each has 100 points
S2	5000	2	100 clusters, each has 50 points

TABLE 2. Summary of the two synthetic datasets.



(A) (B)

FIGURE 1. Two synthetic datasets. The first dataset contains 4 clusters and the second dataset contains 100 clusters.

Since we know the labels of the data points of the two synthetic datasets, we use the corrected Rand index [8, 14, 15, 16] to measure the accuracy of the clustering algorithms. The corrected Rand index, denoted by R_c, ranges from 0 to 1. A higher value of the corrected Rand index indicates a more accurate clustering result.

k	Runtime	R_c
2	0.103(0.139)	0.433(0)
4	0.058(0.114)	1(0)
8	0.106(0.154)	0.682(0.023)

(A) TFCM

k	Runtime	R_c
2	0.044(0.061)	0.498(0)
4	0.05(0.058)	1(0)
8	0.176(0.143)	0.726(0.038)

(B) FCM

TABLE 3. Runtime and accuracy of the TFCM algorithm and the FCM algorithm when applied to the first synthetic dataset 10 times with different initial cluster centers. In the TFCM algorithm, $T = 3$ and other parameters were set to default values given in Table 1. The runtime is measured in seconds. The numbers in the parentheses are the corresponding standard deviations.

Table 3 shows the speed and accuracy of the TFCM algorithm and the FCM algorithm when applied to the first synthetic dataset. Since both algorithms use

random initial cluster centers, we run the two algorithms 10 times to alleviate the impact of initial cluster centers on the performance measures.

The first synthetic dataset contains 400 data points and four clusters of equal size. When the number of clusters k was set to small numbers (e.g., 2 and 4), the TFCM algorithm was slower than the FCM algorithm on average. However, when k was set to 8, the TFCM algorithm outperformed the FCM algorithm on average in terms of speed. In terms of accuracy, both the TFCM algorithm and the FCM algorithm produced a corrected Rand index of 1 when k was set to the true number of clusters. In addition, the standard deviations of the corrected Rand index are zero when $k = 2$ and 4, indicating that the clustering results were not affected by the initial cluster centers. However, when $k = 8$, the standard deviations become positive, indicating that the clustering results were affected by the initial cluster centers.

The test results on the first synthetic dataset show that when k is small, the TFCM algorithm is slower than the FCM algorithm. This is expected as the implementation of TFCM involves sorting distances. The additional runtime caused by sorting is more than the reduced runtime resulted from less number of distance calculations.

k	Runtime	R_c
50	6.869(6.65)	0.502(0.007)
100	5.084(1.97)	0.797(0.029)
200	20.639(7.879)	0.776(0.008)

(A) TFCM with $T = 3$

k	Runtime	R_c
50	5.269(1.574)	0.483(0.007)
100	4.348(1.887)	0.848(0.03)
200	20.184(9.307)	0.777(0.008)

(B) TFCM with $T = 6$

k	Runtime	R_c
50	71.877(16.729)	0.526(0.006)
100	26.341(18.1)	0.819(0.025)
200	53.683(26.543)	0.799(0.015)

(C) FCM

TABLE 4. Runtime and accuracy of the TFCM algorithm and the FCM algorithm when applied to the second synthetic dataset 10 times. The runtime is measured in seconds. The numbers in the parentheses are the corresponding standard deviations.

Table 4 shows the speed and accuracy of the two algorithms when applied to the second synthetic dataset 10 times. The second synthetic dataset contains 5000 data points, which are contained in 100 clusters. Each cluster contains 50 points. Table 4(a) shows the speed and accuracy of the TFCM algorithm when $T = 3$. Comparing Tables 4(a) and 4(c), we see that the TFCM algorithm was significantly faster than the FCM algorithm. For example, it only took the TFCM algorithm about 6.869 seconds to finish clustering the dataset when $k = 50$ and $T = 3$, while it took the FCM algorithm 71.877 seconds to finish clustering the dataset when $k = 50$. For $k = 50, 100$, and 200, the average accuracy of the TFCM algorithm when $T = 3$ is close to that of the FCM algorithm.

If we increased T from 3 to 6, the average accuracy of the TFCM algorithm increased a little bit when $k = 100$ and 200. This is reasonable because when k is

large, increasing T helps find the true cluster centers. Comparing Table 4(b) and Table 4(c), we see that the average corrected Rand index of the TFCM algorithm is higher than that of the FCM algorithm when k was set to the true number of clusters. This might be caused by the fact that we only used 10 runs to calculate the average accuracy. If we use 100 runs to calculate the average corrected Rand index, the FCM algorithm may be more accurate than the TFCM algorithm.

If we look at the average runtime for different k at Tables 4(a), 4(b), and 4(c), we see that the average runtime when k was set to the true number of clusters is lower than that when k was set to other numbers. For example, the average runtime of the TFCM algorithm when $k = 100$ and $T = 3$ was 5.084 seconds, which is lower than the average runtime when $k = 50$ and $k = 200$. We see a similar pattern of runtime for the FCM algorithm. This is caused by the fact that when k is not set to the true number of clusters, it takes both the algorithms more iterations to converge on average.

4.2. **Results on real data.** As we mentioned in the introduction section of this article, data clustering was used to divide a large portfolio of variable annuity contracts into hundreds of clusters in order to find representative contracts for metamodeling [11, 13]. Existing clustering algorithms are slow for dividing a large dataset into hundreds of clusters. In this subsection, we apply the TFCM algorithm to divide a large portfolio of variable annuity contracts into hundreds of clusters.

The variable annuity dataset was simulated by a Java program [12]. The dataset contains 10,000 variable annuity contracts. The original dataset contains categorical variables. We converted the categorical variables into binary dummy variables and normalized all numerical variables to the interval [0,1]. The resulting dataset has 22 numerical features. Since the dataset has no labels, we cannot use the corrected Rand index to measure the accuracy of the clustering results. To compare the clustering results of this dataset, we use the within-cluster sum of squares defined as

$$WSS = \sum_{l=1}^{k} \sum_{\mathbf{x} \in C_l} \sum_{j=1}^{d} (x_j - z_{lj})^2 , \tag{11}$$

where $C_1, C_2, \ldots, C_k$ are k hard clusters obtained from the fuzzy membership matrix U and $\mathbf{z}_l$ is the average of the data points in the cluster C_l. For fixed k, a lower value of WSS indicates a better clustering result.

We applied the TFCM algorithm to this dataset with different values of T. The default value of T for this dataset is 23 because the dimension of this dataset is 22. We also tested the TFCM algorithm with $T = 3, 6, 12$, and 46. The results are shown in Table 5.

Table 5(f) shows the result of the FCM algorithm when applied to the variable annuity dataset. From this table we see that it took the FCM algorithm about 756.378 seconds to divide the dataset into $k = 200$ clusters. The standard deviation of the runtime is also large, indicating that the convergence of the FCM algorithm is sensitive to the initial cluster centers.

Tables 5(a) - 5(e) give the results of the TFCM algorithm when applied to the variable annuity dataset with different values of T. From these tables we see that the runtime of the TFCM algorithm increases when T increases. The tables also show that the TFCM algorithm achieved the best result when $T = 6$. For example, when $T = 6$ and $k = 200$, the within-cluster sum of squares (WSS) produced by the TFCM algorithm is 721.29, which is close to $WSS = 697.841$ produced by

k	Runtime	WSS
100	6.417(1.9)	944.636(14.574)
200	16.167(5.565)	735.001(6.37)

(A) TFCM with $T = 3$

k	Runtime	WSS
100	16.734(5.133)	930.14(15.614)
200	31.871(19.216)	721.291(5.3)

(B) TFCM with $T = 6$

k	Runtime	WSS
100	71.185(22.023)	958.137(15.234)
200	87.918(22.641)	740.548(6.688)

(C) TFCM with $T = 12$

k	Runtime	WSS
100	164.02(57.612)	994.111(18.829)
200	219.695(51.104)	783.113(7.156)

(D) TFCM with $T = 23$

k	Runtime	WSS
100	280.137(70.577)	1049.864(24.202)
200	339.216(80.694)	822.988(8.866)

(E) TFCM with $T = 46$

k	Runtime	WSS
100	597.828(193.2)	895.205(16.264)
200	756.378(382.952)	697.841(6.736)

(F) FCM

TABLE 5. Speed and accuracy of the TFCM algorithm and the FCM algorithm when applied to the variable annuity dataset 10 times. The runtime is in seconds. The numbers in the parentheses are the corresponding standard deviations.

the FCM algorithm. When T increases from 6 to 46, the WSS measure increases. This is counterintuitive because we expect WSS to decrease when T increases. The reason we see that WSS decreased when T increased from 6 to 46 is that we used two criteria to terminate the algorithm: δ and N_{max}. When T is large, we expect that it takes the TFCM algorithm more iterations to get the change of the objective function to be less than δ. Since we terminated the TFCM algorithm when the number of iterations reaches $N_{max} = 1000$, the clustering result was still suboptimal.

If we compare Table 5(b) and Table 5(f), we see that the TFCM algorithm is more than 20 times faster than the FCM algorithm. For example, it took the FCM algorithm about 756 seconds on average to divide the dataset into 200 clusters, but

it only took the TFCM algorithm about 32 seconds on average to divide the dataset into 200 clusters.

In summary, the numerical experiments show that the TFCM algorithm outperformed the FCM algorithm in terms of speed when the desired number of clusters is large. The accuracy of the TFCM algorithm is close to that of the FCM algorithm.

5. **Concluding remarks.** In some situations, we need to divide a large dataset into a large number of clusters. For example, we need to divide millions of web pages into thousands of categories [5] and divide a large portfolio of insurance policies into hundreds of clusters in order to select representative policies [11, 13]. Most existing algorithms are not efficient when used to divide a large dataset into a large number of clusters.

In this paper, we proposed a truncated fuzzy c-means (TFCM) algorithm to address the problem when both the number of data points and the desired number of clusters are large. The TFCM algorithm is similar to the FCM algorithm in the initialization phase. However, the TFCM algorithm is different from the FCM algorithm in the iteration phase where the TFCM algorithm only keeps a subset of cluster centers for each data point and only calculates the distances between each data point and a subset of cluster centers. Our numerical experiments on both synthetic and real datasets show that the TFCM algorithm outperforms the FCM algorithm significantly in terms of speed when the desired number of clusters is large. In addition, the accuracy of the TFCM algorithm is comparable to that of the FCM algorithm.

We implement the TFCM algorithm in a straightforward way according to the pseudo-code given in Algorithm 1. The speed of the TFCM algorithm can be further improved by using the technique introduced by [23]. This technique allows us to combine the step of updating the cluster centers and the step of updating the fuzzy memberships into a single step. In future, we would like to incorporate this technique into the TFCM algorithm and compare the TFCM algorithm with other algorithms mentioned in Section 2.

Acknowledgments. This research was partially supported the National Natural Science Foundation of China (Grant No.71171076).

REFERENCES

[1] C. C. Aggarwal and C. K. Reddy (eds.), *Data Clustering: Algorithms and Applications*, CRC Press, Boca Raton, FL, USA, 2014.

[2] D. Arthur and S. Vassilvitskii, k-means++: The advantages of careful seeding, in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007, 1027–1035.

[3] J. C. Bezdek, R. Ehrlich and W. Full, FCM: The fuzzy c-means clustering algorithm, *Computers & Geosciences*, **10** (1984), 191–203.

[4] J. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[5] A. Broder, L. Garcia-Pueyo, V. Josifovski, S. Vassilvitskii and S. Venkatesan, Scalable k-means by ranked retrieval, in *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, ACM, 2014, 233–242.

[6] R. L. Cannon, J. V. Dave and J. Bezdek, Efficient implementation of the fuzzy c-means clustering algorithms, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-8** (1986), 248–255.

[7] T. W. Cheng, D. B. Goldgof and L. O. Hall, Fast fuzzy clustering, *Fuzzy Sets and Systems*, **93** (1998), 49–56.

[8] M. de Souto, I. Costa, D. de Araujo, T. Ludermir and A. Schliep, Clustering cancer gene expression data: A comparative study, *BMC Bioinformatics*, **9** (2008), p497.
[9] J. C. Dunn, A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters, *Journal of Cybernetics*, **3** (1973), 32–57.
[10] G. Gan, *Data Clustering in C++: An Object-Oriented Approach*, Data Mining and Knowledge Discovery Series, Chapman & Hall/CRC Press, Boca Raton, FL, USA, 2011.
[11] G. Gan, Application of data clustering and machine learning in variable annuity valuation, *Insurance: Mathematics and Economics*, **53** (2013), 795–801.
[12] G. Gan, A multi-asset Monte Carlo simulation model for the valuation of variable annuities, in *Proceedings of the Winter Simulation Conference*, 2015, 3162–3163.
[13] G. Gan and S. Lin, Valuation of large variable annuity portfolios under nested simulation: A functional data approach, *Insurance: Mathematics and Economics*, **62** (2015), 138–150.
[14] G. Gan and M. K.-P. Ng, Subspace clustering using affinity propagation, *Pattern Recognition*, **48** (2015), 1455–1464.
[15] G. Gan and M. K.-P. Ng, Subspace clustering with automatic feature grouping, *Pattern Recognition*, **48** (2015), 3703–3713.
[16] G. Gan, Y. Zhang and D. K. Dey, Clustering by propagating probabilities between data points, *Applied Soft Computing*, **41** (2016), 390–399.
[17] R. J. Hathaway and J. C. Bezdek, Extending fuzzy and probabilistic clustering to very large data sets, *Computational Statistics & Data Analysis*, **51** (2006), 215–234.
[18] T. Havens, J. Bezdek, C. Leckie, L. Hall and M. Palaniswami, Fuzzy c-means algorithms for very large data, *IEEE Transactions on Fuzzy Systems*, **20** (2012), 1130–1146.
[19] M.-C. Hung and D.-L. Yang, An efficient fuzzy c-means clustering algorithm, in *Proceedings IEEE International Conference on Data Mining*, 2001, 225–232.
[20] Z.-X. Ji, Q.-S. Sun and D.-S. Xia, A modified possibilistic fuzzy c-means clustering algorithm for bias field estimation and segmentation of brain MR image, *Computerized Medical Imaging and Graphics*, **35** (2011), 383–397.
[21] D. Jiang, C. Tang and A. Zhang, Cluster analysis for gene expression data: A survey, *IEEE Transactions on Knowledge and Data Engineering*, **16** (2004), 1370–1386.
[22] F. Klawonn, Fuzzy clustering: Insights and a new approach, *Mathware & Soft Computing*, **11** (2004), 125–142.
[23] J. F. Kolen and T. Hutcheson, Reducing the time complexity of the fuzzy c-means algorithm, *IEEE Transactions on Fuzzy Systems*, **10** (2002), 263–267.
[24] T. Kwok, K. Smith, S. Lozano and D. Taniar, Parallel fuzzy c- means clustering for large data sets, in *Euro-Par 2002 Parallel Processing* (eds. B. Monien and R. Feldmann), vol. 2400 of Lecture Notes in Computer Science, Springer, 2002, 365–374.
[25] H. Liu, F. Zhao and L. Jiao, Fuzzy spectral clustering with robust spatial information for image segmentation, *Applied Soft Computing*, **12** (2012), 3636–3647.
[26] J. D. MacCuish and N. E. MacCuish, *Clustering in Bioinformatics and Drug Discovery*, CRC Press, Boca Raton, FL, 2010.
[27] J. Macqueen, Some methods for classification and analysis of multivariate observations, in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics andProbability* (eds. L. LeCam and J. Neyman), University of California Press, Berkely, CA, USA, **1** (1967), 281–297.
[28] S. A. A. Shalom, M. Dash and M. Tue, Graphics hardware based efficient and scalable fuzzy c-means clustering, in *Proceedings of the 7th Australasian Data Mining Conference*, **87** (2008), 179–186.
[29] A. Stetco, X.-J. Zeng and J. Keane, Fuzzy c-means++: Fuzzy c-means with effective seeding initalization, *Expert Systems with Applications*, **42** (2015), 7541– 7548.

Received July 2016; revised August 2016.

E-mail address: guojun.gan@uconn.edu
E-mail address: lanqiujun@hnu.edu.cn
E-mail address: shiyang.sima0222@gmail.com