JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 27, 995-1009 (2011)

A Fuzzy K-means Clustering Algorithm Using Cluster Center Displacement

CHIH-TANG CHANG[1], JIM Z. C. LAI[2] AND MU-DER JENG[1]
[1]*Department of Electrical Engineering*
[2]*Department of Computer Science and Engineering*
National Taiwan Ocean University
Keelung, 202 Taiwan

In this paper, we present a fuzzy *k*-means clustering algorithm using the cluster center displacement between successive iterative processes to reduce the computational complexity of conventional fuzzy *k*-means clustering algorithm. The proposed method, referred to as CDFKM, first classifies cluster centers into active and stable groups. Our method skips the distance calculations for stable clusters in the iterative process. To speed up the convergence of CDFKM, we also present an algorithm to determine the initial cluster centers for CDFKM. Compared to the conventional fuzzy *k*-means clustering algorithm, our proposed method can reduce computing time by a factor of 3.2 to 6.5 using the data sets generated from the Gauss Markov sequence. Our algorithm can reduce the number of distance calculations of conventional fuzzy *k*-means clustering algorithm by 38.9% to 86.5% using the same data sets.

Keywords: vector quantization, fuzzy *k*-means clustering, data clustering, knowledge discovery, pattern recognition

1. INTRODUCTION

Data clustering is used frequently in a number of applications, such as vector quantization (VQ) [1-4], pattern recognition [5], knowledge discovery [6], speaker recognition [7], fault detection [8], and web/data mining [9]. Among clustering formulations that minimize an objective function, fuzzy *k*-means clustering is widely used and studied [10]. The fuzzy *k*-means clustering algorithm is a special case of the generalized fuzzy *k*-means clustering scheme, where point representatives are adopted and the Euclidean distance is used to measure the dissimilarity between a vector **X** and its cluster representative **C**.

The fuzzy *k*-means clustering (FKM) algorithm performs iteratively the partition step and new cluster representative generation step until convergence. The applications of FKM can be founded in reference [11], which provided an excellent review of FKM. An iterative process with extensive computations is usually required to generate a set of cluster representatives. The convergence of FKM is usually much lower than that of hard *k*-means clustering [12]. Some methods are available to speed up hard *k*-means clustering [13-15]. Kanungo *et al.* [13] developed a filtering algorithm on a kd-tree to speed up the generation of new cluster centers. Sorting data points in a kd-tree for *k*-means clustering was also used by Pelleg and Moore [14]. After some iterations of hard *k*-means clustering, most of the centers are converged to their final positions and the majority of data points

Received September 14, 2009; revised January 19 & March 24, 2010; accepted September 7, 2010.
Communicated by Chih-Jen Lin.

have few candidates to be selected as their closest centers. Lai *et al.* [15] exploited this characteristic to develop a fast *k*-means clustering algorithm to reduce the computational complexity of *k*-means clustering.

To reduce the computational complexity of FKM, Shankar and Pal used multistage random sampling to reduce the data size [16]. This method reduced the computational complexity by a factor of 2 to 4. Cannon, Dave, and Bezdek used look-up tables for storing distances to approximate fuzzy *k*-mean clustering and reduced the computing time by a factor of about 6 [17]. It is noted that this method is applicable only for integer-valued data in the range of 0 to 255 and the accuracy of a cluster center's coordinate is up to 0.1. Höppner developed an approximate FKM to reduce the computational complexity of conventional FKM [18]. This method gave the same membership as that of conventional FKM within a given precision and reduced the computing time of conventional FKM by a factor of 2 to 4. It is noted that all the above method cannot obtain the same clustering result as that of conventional FKM. After some iterations of FKM, it is expected that many of the centers are converged to their final positions and many distance calculations can be avoided at each partition step. This characteristic is exploited to reduce the computational complexity of fuzzy *k*-means clustering.

In this paper, two algorithms are presented to reduce the computing time of fuzzy *k*-means clustering. These two algorithms classify cluster centers (representatives) into stable and active groups and the distance calculations are executed only for those active cluster representatives during the iterative process. This paper is organized as follows. Section 2 describes the fuzzy *k*-means clustering algorithm. Section 3 presents the algorithms developed in this paper. Some theoretical analyses of the presented algorithms are also shown in section 3. Experimental results are presented in section 4 and concluding remarks are given in section 5.

2. FUZZY K-MEANS CLUSTERING ALGORITHM

The fuzzy *k*-means clustering algorithm partitions data points into *k* clusters S_l (l = 1, 2, …, k) and clusters S_l are associated with representatives (cluster center) $\mathbf{C}_l$. The relationship between a data point and cluster representative is fuzzy. That is, a membership $u_{i,j} \in [0, 1]$ is used to represent the degree of belongingness of data point $\mathbf{X}_i$ and cluster center $\mathbf{C}_j$. Denote the set of data points as $S = \{\mathbf{X}_i\}$. The FKM algorithm is based on minimizing the following distortion:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{N} u_{i,j}^{m} d_{ij} \tag{1}$$

with respect to the cluster representatives $\mathbf{C}_j$ and memberships $u_{i,j}$, where N is the number of data points; m is the fuzzifier parameter; k is the number of clusters; and d_{ij} is the squared Euclidean distance between data point $\mathbf{X}_i$ and cluster representative $\mathbf{C}_j$. It is noted that $u_{i,j}$ should satisfy the following constraint:

$$\sum_{j=1}^{k} u_{i,j} = 1, \text{ for } i = 1 \text{ to } N. \tag{2}$$

The major process of FKM is mapping a given set of representative vectors into an improved one through partitioning data points. It begins with a set of initial cluster centers and repeats this mapping process until a stopping criterion is satisfied. It is supposed that no two clusters have the same cluster representative. In the case that two cluster centers coincide, a cluster center should be perturbed to avoid coincidence in the iterative process. If $d_{ij} < \eta$, then $u_{i,j} = 1$ and $u_{i,l} = 0$ for $l \neq j$, where η is a very small positive number. The fuzzy *k*-means clustering algorithm is now presented as follows.

(1) Input a set of initial cluster centers $SC_0 = \{\mathbf{C}_j(0)\}$ and the value of ε. Set $p = 1$.
(2) Given the set of cluster centers SC_p, compute d_{ij} for $i = 1$ to N and $j = 1$ to k. Update memberships $u_{i,j}$ using the following equation:

$$u_{i,j} = \left((d_{ij})^{1/m-1} \sum_{l=1}^{k} \left(\frac{1}{d_{il}} \right)^{1/m-1} \right)^{-1}. \tag{3}$$

If $d_{ij} < \eta$, set $u_{i,j} = 1$, where η is a very small positive number.
(3) Compute the center for each cluster using Eq. (4) to obtain a new set of cluster representatives SC_{p+1}.

$$\mathbf{C}_j(p) = \frac{\sum_{i=1}^{N} u_{ij}^m \mathbf{X}_i}{\sum_{i=1}^{N} u_{ij}^m} \tag{4}$$

(4) If $\|\mathrm{C}_j(p) - \mathrm{C}_j(p-1)\| < \varepsilon$ for $j = 1$ to k, then stop, where $\varepsilon > 0$ is a very small positive number. Otherwise set $p + 1 \rightarrow p$ and go to step 2.

The major computational complexity of FKM is from steps 2 and 3. However, the computational complexity of step 3 is much less than that of step 2. Therefore the computational complexity, in terms of the number of distance calculations, of FKM is $O(Nkt)$, where t is the number of iterations.

3. PROPOSED METHODS

In the iterative process of fuzzy *k*-means clustering, one may expect that the displacements of some cluster centers will be smaller than the threshold ε after few times of iterations and others need the much longer times of iterations to be stabilized, where $\varepsilon > 0$ is a very small positive number. Let the jth cluster centers used in the current and previous partitions be denoted as $\mathbf{C}_j$ and $\mathbf{C}'_j$, respectively. Denote the displacement between $\mathbf{C}_j$ and $\mathbf{C}'_j$ as D_j. That is, $D_j = \|\mathbf{C}_j - \mathbf{C}'_j\|$. If $D_j < \varepsilon$, then the vector $\mathbf{C}_j$ is defined as a stable cluster center; otherwise it is called an active cluster center. The cluster associated with an active center is called an active cluster. Similarly the cluster having a stable center is defined as a stable cluster. The number of stable cluster centers increases as the iteration proceeds [19].

3.1 Fuzzy K-means Clustering Algorithm Using Cluster Displacement

Denote the subsets, which consist of active cluster centers and stable cluster centers as SC_a and SC_s, respectively. Let $k_{a,i}$ be the number of clusters in SC_a at the ith iteration of fuzzy k-means clustering. The value of $k_{a,i}$ decreases as the iteration proceeds in general. The performance, in terms of computing time, of the proposed method is better, if $k_{a,i}$ decreases more quickly during the process of iteration. The value and creasing rate of $k_{a,i}$ depend on data distribution. For a data set with good data separation, $k_{a,i}$ will decrease quickly. For an evenly distributed data set, $k_{a,i}$ will decrease slowly. For a real data set, a good data separation is usually obtained. In the worst case, $k_{a,i}$ equals k, which is the number of clusters. It is noted that centers of clusters in the previous iteration will be used to partition the set of data points $\{\mathbf{X}_i\}$ in the current iteration. The FKM algorithm stops, if the displacements of all cluster centers are less than ε. That is, if $D_j < \varepsilon$, then cluster S_j is a stable cluster and d_{ij} (i = 1 to N) will not be recalculated to update $u_{i,j}$ in the iterative process. The proposed algorithm will use this property to speed up fuzzy k-means clustering. Now, the fuzzy k-means clustering algorithm using cluster displacement (CDFKM) is presented below.

Fuzzy *K*-means Clustering Algorithm Using Center Displacement

(1) Input a set of initial cluster centers SC_0 and the value of ε. Set $p = 1$.
(2) Given the set of cluster centers $SC_0 = \{\mathbf{C}_j(0)\}$, compute d_{ij} for i = 1 to N and j = 1 to k. Use Eq. (3) to update $u_{i,j}$. If $d_{ij} < \eta$, set $u_{i,j} = 1$.
(3) Use Eq. (4) to update $\mathbf{C}_j(p)$ and calculate $D_j = \|C_j(p) - C_j(p-1)\|$ for j = 1 to k. Set q = 1.
(4) If $D_q < \varepsilon$, go to step 5; otherwise compute d_{iq} for i = 1 to N.
(5) Set $q = q + 1$. If $q > k$ go to step 6; otherwise go to step 4.
Use Eq. (3) to compute $u_{i,j}$ (if $d_{ij} < \eta$, set $u_{i,j} = 1$) for i = 1 to N and j = 1 to k.
(6) Set $p = p + 1$, use Eq. (4) to update $\mathbf{C}_j(p)$, and calculate $D_j = \|C_j(p) - C_j(p-1)\|$ for j = 1 to k. If $D_j < \varepsilon$ for j = 1 to k, then stop; otherwise set q = 1 and go to step 4.

3.2 Parameter Selection and Computational Complexity Analysis

In this subsection, the effect of ε on $u_{i,j}$ will be investigated. Eq. (3) can be rewritten as

$$u_{i,j} = \frac{1}{1 + \sum_{l=1, l\neq j}^{k} \left(\frac{d_{ij}}{d_{il}} \right)^{1/m-1}}. \tag{5a}$$

That is,

$$\sum_{l=1, l\neq j}^{k} \left(\frac{d_{ij}}{d_{il}} \right)^{1/m-1} = \left(\frac{1}{u_{i,j}} - 1\right). \tag{5b}$$

Let the squared Euclidean distances between data point $\mathbf{X}_i$ and cluster centers $C_j(p -$

1) and $C_j(p)$ be d'_{ij} and d_{ij}, respectively. In the case of $\|C_j(p) - C_j(p-1)\| < \varepsilon$, d'_{ij} is used to calculate memberships $u_{i,j}$. Denote $u'_{i,j}$ as the membership of $\mathbf{X}_i$ with respect to $C_j(p)$, if d_{ij} is replaced by d'_{ij} in Eq. (3). Similarly, let $u_{i,j}$ be the membership of $\mathbf{X}_i$ with respect to $C_j(p)$ for the case that d_{ij} is used to calculate memberships. For many applications, $m = 2$ is used [10] and is adopted in this paper. In the case of $\|C_j(p) - C_j(p-1)\| < \varepsilon$, d'_{ij} can be estimated by

$$d'_{ij} \approx d'_{ij} \pm O(\varepsilon)(d'_{ij})^{1/2}. \tag{6}$$

Eq. (6) implies that if $\|C_j(p) - C_j(p-1)\| < \varepsilon$, then $|d'_{ij} - d_{ij}|/(d_{ij})^{1/2} \approx O(\|C_j(p) - C_j(p-1)\|)$. That is, $|d'_{ij} - d_{ij}|/(d_{ij})^{1/2}$ equals approximately the displacement of the ith cluster's center. $u'_{i,j}$ is obtained by replacing d_{ij} in Eq. (5a) by d'_{ij} given in Eq. (6). That is,

$$\begin{aligned} u'_{i,j} &\approx \frac{1}{1 + \sum_{l=1,l\neq j}^{k} \left(\frac{d_{ij}}{d_{il}}\right) \pm O(\varepsilon)(d_{ij})^{1/2} \sum_{l=1,l\neq j}^{k} \frac{1}{d_{il}}} \\ &= \frac{1}{1 + \sum_{l=1,l\neq j}^{k} \left(\frac{d_{ij}}{d_{il}}\right) \pm O(\varepsilon)(d_{ij})^{-1/2} \sum_{l=1,l\neq j}^{k} \frac{d_{ij}}{d_{il}}}, \end{aligned} \tag{7a}$$

for $m = 2$. For the case of $m = 2$, Eq. (5b) becomes

$$\sum_{l=1,l\neq j}^{k} \left(\frac{d_{ij}}{d_{il}}\right) = \left(\frac{1}{u_{i,j}} - 1\right). \tag{7b}$$

Substituting the term $\sum_{l=1,l\neq j}^{k} \left(\frac{d_{ij}}{d_{il}}\right)$ in Eq. (7a) by $\left(\frac{1}{u_{i,j}} - 1\right)$ as shown in Eq. (7b) gives

$$u'_{i,j} \approx \frac{1}{\frac{1}{u_{i,j}} \pm O(\varepsilon)(d_{ij})^{-1/2} \left(\frac{1}{u_{i,j}} - 1\right)} \approx u'_{i,j} \pm O(\varepsilon)(d_{ij})^{1/2}(u'_{i,j} - (u'_{i,j})^2). \tag{8}$$

That is, $|u'_{i,j} - u_{i,j}| \approx O(\varepsilon)(d_{ij})^{-1/2}(u_{i,j} - (u_{i,j})^2) \le O(\varepsilon)(d_{ij})^{-1/2}$. In the case of $d_{ij} < \eta$, it implies that $u_{i,j} = u'_{i,j} = 1$ and $|u'_{i,j} - u_{i,j}| = 0$. Since η is a very small positive number, it implies that $\eta << \eta^{1/2}$. In the case of $\eta \le d_{ij} \le \varepsilon$, one can obtain $|u'_{i,j} - u_{i,j}| \le O(\varepsilon)/\eta^{1/2}$. If $\varepsilon = \eta$ is chosen, $|u'_{i,j} - u_{i,j}| << 1$ will be obtained due to $\eta << \eta^{1/2}$. In this paper, $\varepsilon = \eta = 0.00001$ is used.

The major computational complexity of CDFKM is from steps 4, 6, and 7. To compute $u_{i,j}$, $\sum_{l=1}^{k} \left(\frac{1}{d_{il}}\right)^{1/m-1}$ for each data point $\mathbf{X}_i$ are first calculated and stored. That is, Nk multiplications and additions are needed to update $u_{i,j}$ at step 6. To calculate distances between all data points and cluster centers, Nk distance calculations are required. Each distance calculation requires d multiplications and $(2d - 1)$ additions, where d is the data

dimension. That is, Nkd multiplications and $Nk(2d - 1)$ additions are needed to calculate distances at step 4. Therefore, it can be concluded that the computational complexities of steps 6 and 7 are much less than that of step 4. Let k_a be the average number of active cluster centers at each stage of iteration for CDFKM, where $k_a = (\sum_{i=0}^{t-1} k_{a,i})/t$, t is the number of iteration, and $k_{a,i}$ is the number of active clusters at the ith iteration. For a data set with good data separation, a small value of k_a is expected. For an evenly distributed data set, k_a may equal k. For the worst case, k_a equals k and the proposed method will make no improvement over original FKM. The probability of performing distance calculations at step 4 is k_a/k, where k is the number of clusters. That is, the computational complexity, in terms of the number of distance calculations, of CDFKM is $O(Nkt) \times O(k_a/k) = O(Nk_at)$, where t is the number of iterations. Since $1 \le k_a \le k$, the computational complexity, in terms of the number of distance calculations, of CDFKM is upper bounded by $O(Nkt)$.

3.3 Determination of Initial Cluster Centers Using Subsets

The computing time and number of iterations for CDFKM increase as the data size increases. The proposed method first uses CDFKM to generate a codebook from subsets of S and then adopt this codebook as the initial codebook for CDFKM to partition the whole data set S into k clusters. This initial approximation helps to reduce the number of iterations for CDFKM. To speed up the convergence of CDFKM, M subsets of the data set S are used to estimate the initial cluster centers for CDFKM. Denote these M subsets as SB_l ($l = 1$ to M). The data size of SB_l is fN, where $f < 1$ and N is the data points in S. It is noted that the data points in SB_l are selected randomly from S and $SB_i \cap SB_j = \varnothing$ ($i \neq j$), where $\varnothing$ is an empty set. The cluster center estimation algorithm (CCEA) first generates an initial set of cluster centers SC_0, which is obtained by selecting randomly k data points from SB_1, for CDFKM to partition SU, where $SU = SB_1$. This partition process will generate a set of cluster centers SC_1. Setting $SU = SU \cup SB_p$, where $p = 2$ to M, CCEA uses SC_{p-1} as the initial set of cluster centers for CDFKM to generate SC_p using the data set SU. This process is repeated until the set of cluster centers SC_M is obtained. Finally, CDFKM uses SC_M as the initial set of cluster centers to partition the whole data set S. Now, the cluster center estimation algorithm (CCEA) is presented as follows,

Cluster Center Estimation Algorithm

(1) Randomly select M subsets SB_l ($l = 1$ to M) of size fN from the data set S such that $SB_i \cap SB_j = \varnothing$ ($i \neq j$), where $f < 1$. Set $SU = SB_1$ and $p = 0$.
(2) Given a set of initial cluster centers $SC_p = \{\mathbf{C}_j\}$ and the data set SU, use CDFKM to determine a set of cluster centers $SC_{p+1} = \{\mathbf{C}_j\}$.
(3) Update $p = p + 1$ and set $SU = SU \cup SB_{p+1}$. If $p \le M$, go to step 2.
(4) Output SC_M as the set of initial cluster centers.

The set SC_0 is obtained by selecting randomly k data points from the subset SB_1. A subset of size sN is used by CCEA to obtain an initial set of cluster centers for CDFKM, where $0 < s < 1$. Note that f is a small positive real number. The value of M is so chosen that it is less than (s/f). Using the initial cluster centers determined by CCEA for CDFKM, the corresponding algorithm is denoted as modified CDFKM (MCDFKM).

4. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed algorithms CDFKM and MCDFKM, several sets of synthetic data, a set of real images, and a real data set have been used. The values of f and ε for MCDFKM are set to be 0.05 and 0.00001, respectively. In the first example, the data set has about 50,000 data points. This data set is obtained from real images consists of image blocks of 4×4 pixels. That is, the set of data points with dimension 16 is obtained from three real images: "Peppers," "Lena," and "Baboon." It is noted that the block size for each data point is 4×4. In example 2, several data sets with size 20,000 and dimensions from 8 to 40 are generated. There are 40 cluster centers, which are evenly distributed over the hypercube $[-1, 1]^d$ with d ranging from 8 to 40. A Gaussian distribution with standard distribution $\sigma = 0.05$ along each coordinate is used to generate data points around each center, where each coordinate is generated independently. In example 3, several data sets with size 10,000 and dimensions from 8 to 40 are obtained from the Gauss Markov sequence [1] with $\sigma = 10$, $\mu = 0$, and $a = 0.9$, where σ is the standard deviation, μ is the mean value of the sequence and a is the correlation coefficient. In example 4, the Statlog (Landsat Satellite) data set consisting of 6,435 data points with 36 attributes is used [20].

In these examples, the proposed algorithms are compared to the conventional fuzzy k-means clustering algorithm in terms of the average number of distance calculations and computing time. The average computing time and number of distance calculations are calculated for each algorithm with 50 repetitions using different initial cluster centers. The initial cluster centers are randomly selected from each data set. Every algorithm uses the same initial cluster centers at each repetition. All computing is performed on an AMD Dual Opteron 2.0 GHz PC with 2GB of memory. All programs are implemented as console applications of Microsoft Visual Studio 6.0 and are executed under Windows XP Professional SP3.

Example 1: A data set is generated from three real images.

In the first example, this data set with $d = 16$ and $N \approx 50{,}000$ is generated from three real images: "Lena," "Peppers," and "Baboon." Tables 1 and 2 give the average execution time and number of distance calculations per data point, respectively, for FKM (fuzzy k-means clustering algorithm), CDFKM, and MCDFKM. Table 3 presents the average distortion per data point for these three methods. From Tables 1 and 2, it can be concluded that MCDFKM with $M = 1$ has the best performance, in terms of the computing time and number of distance calculations, for $k \leq 32$. For $k \geq 64$, CDFKM has the least computing time and number of distance calculations. Compared to FKM, CDFKM can reduce the computing time by a factor of 1.1 to 2.1. From Table 1, it can be found that the computing time of MCDFKM increases as the value of M increases. Therefore MCDFKM with $M = 1$ is used in the following examples. From Table 3, it can be found that FKM, CDFKM, and MCDFKM can obtain the same clustering result.

Example 2: Data sets are generated with cluster centers evenly distributed over the hypercube $[-1, 1]^d$.

In this example, each data set consists of 20,000 data points. Fig. 1 shows the average computing time for FKM, CDFKM, and MCDFKM with $M = 1$, whereas Fig. 2 gives

Table 1. The average computing time (in seconds) for FKM, CDFKM, and MCDFKM using a data set generated three real images.

Method	k			
	16	32	64	128
FKM	509.50	1328.92	2925.56	15355.94
CDFKM	468.86	1212.73	2329.64	7920.42
MCDFKM (M = 1)	338.67	894.00	3192.59	7623.59
MCDFKM (M = 2)	349.08	911.80	3434.36	7791.73
MCDFKM (M = 3)	394.97	983.69	3678.59	8404.13
MCDFKM (M = 4)	412.97	1072.31	4069.98	10147.78
MCDFKM (M = 5)	475.51	1138.83	4390.61	10205.95
MCDFKM (M = 6)	530.64	1294.08	5385.81	14342.30

Table 2. The average number of distance calculations per data point for FKM, CDFKM, and MCDFKM using a data set generated from three real images.

Method	k			
	16	32	64	128
FKM	82577024	210768032	462431360	2415968128
CDFKM	65275360	173313824	225074624	398747520
MCDFKM (M = 1)	41237844	98798730	289202130	462964383
MCDFKM (M = 2)	42805470	106209666	301687848	503016639
MCDFKM (M = 3)	47960496	112179816	309157617	525194391
MCDFKM (M = 4)	49906440	118076676	341583657	555601119
MCDFKM (M = 5)	56393328	123518739	360170358	623380608
MCDFKM (M = 6)	62447400	136084641	387006372	669698964

Table 3. The average distortion per data point for FKM, CDFKM, and MCDFKM using a data set generated from three real images.

Method	k			
	16	32	64	128
FKM	828.68	404.38	200.13	99.76
CDFKM	828.68	404.38	200.13	99.76
MCDFKM (M = 1)	828.68	404.38	200.13	99.76
MCDFKM (M = 2)	828.68	404.38	200.13	99.76
MCDFKM (M = 3)	828.69	404.38	200.13	99.77
MCDFKM (M = 4)	828.69	404.38	200.13	99.77
MCDFKM (M = 5)	828.69	404.38	200.13	99.77
MCDFKM (M = 6)	828.69	404.38	200.13	99.77

the average number of distance calculations per data point for FKM, CDFKM, and MCDFKM. Fig. 3 shows the average distortion per data point for these three methods. From Figs. 1 and 2, it can be found that the proposed method MCDFKM with M = 1 outperforms FKM in terms of the computing time and number of distance calculations. From Fig. 1, it can also be found find that the computing time of the proposed approach MCDFKM with M = 1 grows linearly with data dimension. Compared to FKM, MCDFKM with M = 1 can reduce the computing time by a factor of 2.6 to 3.1. Fig. 3 shows that FKM, CDFKM, and MCDFKM with M = 1 can obtain the same clustering result.

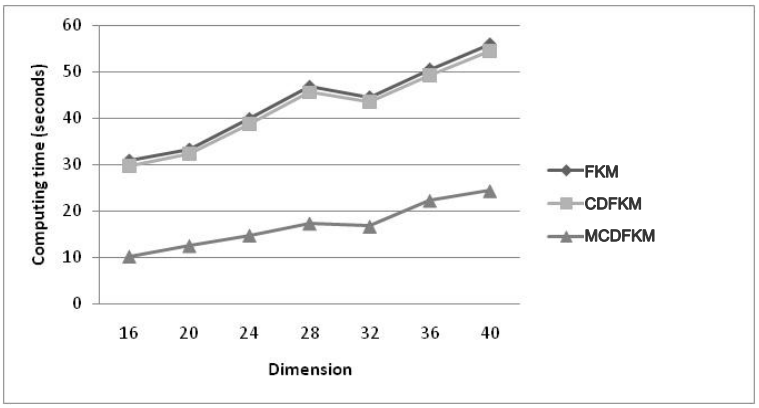

Fig. 1. The average computing time for data sets from example 2 with $N = 20{,}000$ and $k = 40$.

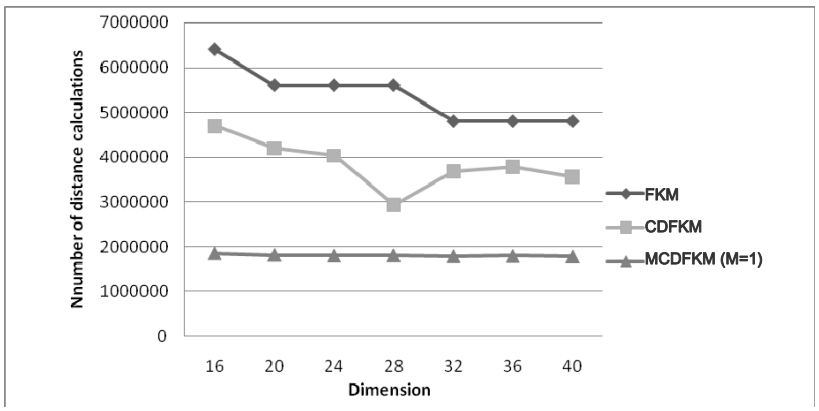

Fig. 2. The average number of distance calculations per data point for data sets from example 2 with $N = 20{,}000$ and $k = 40$.

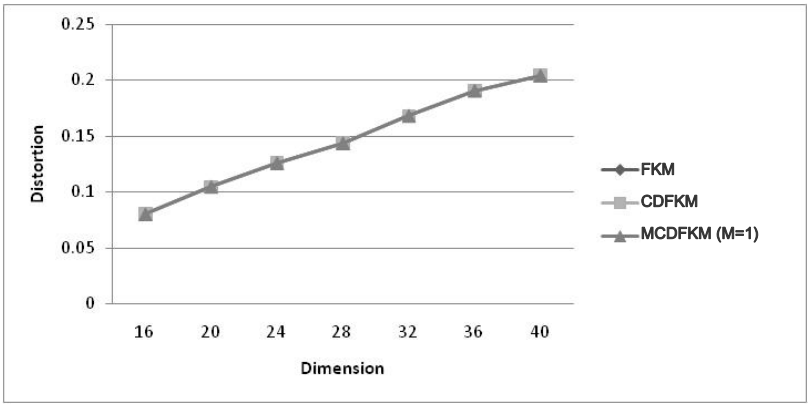

Fig. 3. The average distortion per data point for data sets from example 2 with $N = 20{,}000$ and $k = 40$.

Example 3: Data sets are generated from the Gauss Markov sequence.

In this example, each data set is obtained from the Gauss Markov source. Figs. 4 and 7 present the average computing time with $k = 128$ and 256, respectively. Figs. 5 and 8 show the average number of distance calculations per data point with $k = 128$ and 256, respectively. Figs. 6 and 9 present the average distortion per data point. From these figures, it can be found that the computing time of MCDFKM with $M = 1$ increases linearly with the data dimension d. From Figs. 4, 5, 7, and 8, it can be found that MCDFKM with $M = 1$ has the best performance in terms of the computing time and number of distance calculations for all cases. Compared with FKM, the proposed method MCDFKM with $M = 1$ can reduce the computing time by a factor of 3.0 to 6.5. From Figs. 5 and 8, it can be found that the number of distance calculations of the proposed approach MCDFKM with $M = 1$ is independent of data dimension. Figs. 6 and 9 also confirm that the proposed approaches and FKM can obtain the same clustering result.

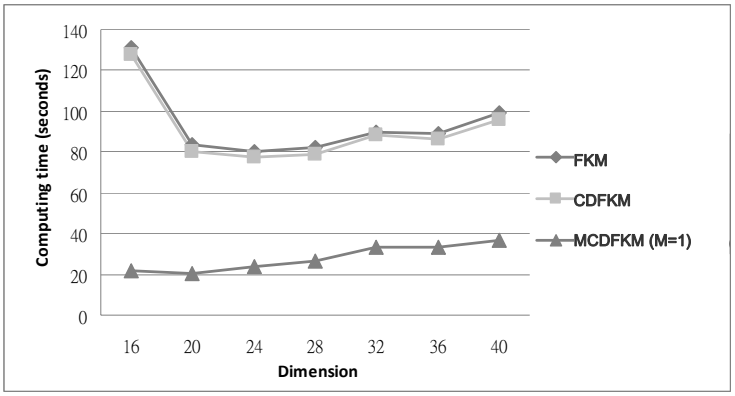

Fig. 4. The average computing time for data sets from example 3 with $N = 10{,}000$ and $k = 128$.

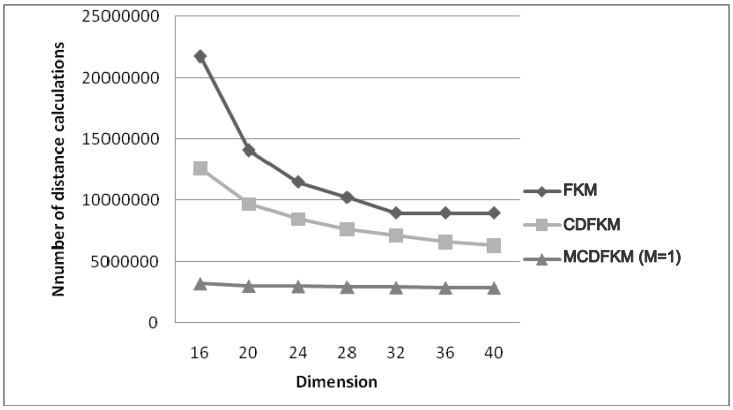

Fig. 5. The average number of distance calculation per data point for data sets from example 3 with $N = 10{,}000$ and $k = 128$.

Example 4: The Statlog (Landsat Satellite) data set.

In the fourth example, the Statlog (Landsat Satellite) data set consists of 6,435 data points with $d = 36$ [20]. The value of each coordinate for a data point is in the range of 0 to 255. The more detailed description regarding this data set can be found in [20]. Tables 4 and 5 present the computing time and number of distance calculations per data point, respectively, for three algorithms to partition this data set into 16, 32, and 64 clusters. Table 6 gives the average distortion per data point. From Tables 4 and 5, it can be concluded that MCDFKM with $M = 1$ has the best performance in terms of computing time and number of distance calculations. Compared to FKM, CDFKM can reduce the computing time by a factor of 1.58 to 2.47. From Table 6, it can be found that FKM, CDFKM, and MCDFKM can obtain the same clustering result.

From Examples 1 and 4, it is found that CDFKM has the better performance for real data sets. Since real data sets usually have the high cluster separation, it is recommended that CDFKM in stead of FKM is used for a data set with high cluster separation. It is noted that CDFKM is a fast version of FKM.

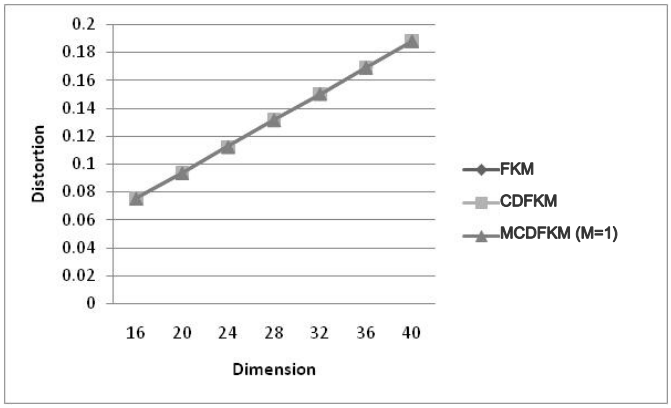

Fig. 6. The average distortion per data point for data sets from example 3 with $N = 10{,}000$ and $k = 128$.

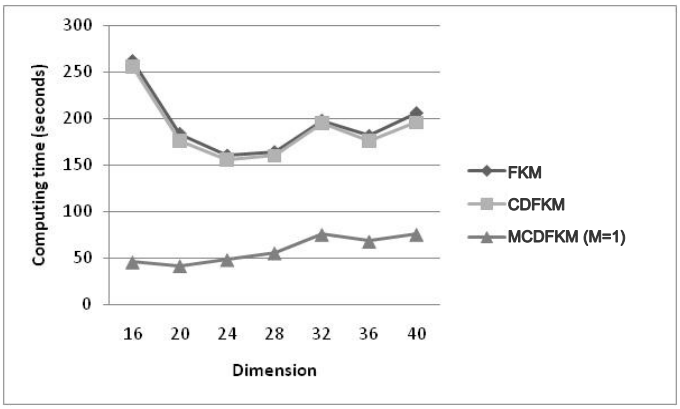

Fig. 7. The average computing time for data sets from example 3 with $N = 10{,}000$ and $k = 256$.

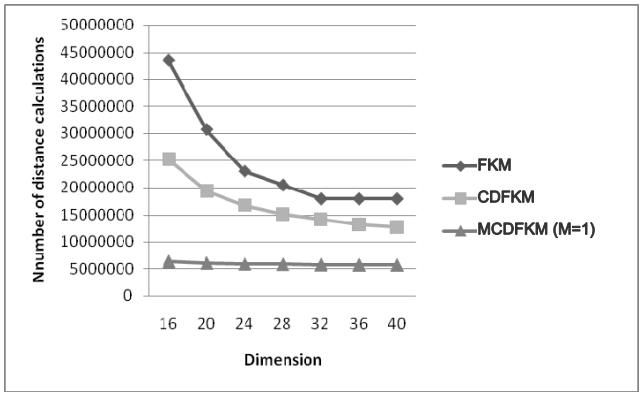

Fig. 8. The average number of distance calculation per data point for data sets from example 3 with N = 10,000 and k = 256.

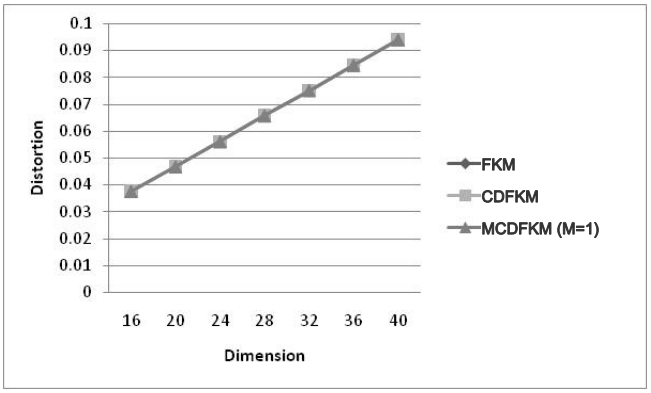

Fig. 9. The average distortion per data point for data sets from example 3 with N = 10,000 and k = 256.

Table 4. The average computing time (in seconds) for FKM, CDFKM, and MCDFKM using the Statlog data set.

Method	k		
	16	32	64
FKM	297.64	916.06	1838.70
CDFKM	187.81	371.50	802.89
MCDFKM (M = 1)	143.64	314.49	724.46

To visualize the clustering result, a data set with N = 1000 and d = 2 is generated. This data set has ten cluster centers distributed over the hypercube $[-1, 1]^2$. A Gaussian distribution with standard deviation = 0.15 along each coordinate is used to generate data points around each center. This data set is divided into 10 clusters using FKM and MCDFKM with M = 1. Fig. 10 presents the clustering results. From Fig. 10, it can be found that FKM and MCDFKM with M = 1 can obtain the same clustering result.

Table 5. The average number of distance calculations per data point for FKM, CDFKM, and MCDFKM using the Statlog data set.

Method	k		
	16	32	64
FKM	25743984	79291488	158171072
CDFKM	9706652	17309091	29728478
MCDFKM ($M = 1$)	9855873	18970822	40250065

Table 6. The average distortion per data point for FKM, CDFKM, and MCDFKM using the Statlog data set.

Method	k		
	16	32	64
FKM	1368.78	684.39	342.19
CDFKM	1368.78	684.39	342.19
MCDFKM ($M = 1$)	1368.78	684.39	342.19

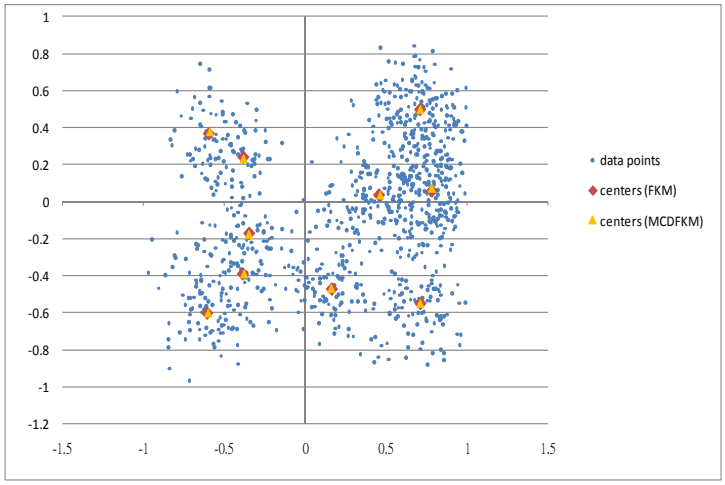

Fig. 10. The clustering results for FKM and MCDFKM with $M = 1$ using a synthetic data set of size 1000.

5. CONCLUSIONS

In this paper, two novel algorithms are developed to speed up fuzzy k-means clustering through using the information of center displacement between two successive partition processes. A cluster center estimation algorithm is also presented to determine the initial cluster centers for the proposed algorithm CDFKM. The computing time of the proposed algorithm MCDFKM with $M = 1$ grows linearly with data dimension Compared to FKM, the proposed algorithm MCDFKM with $M = 1$ can effectively reduce the computing time and number of distance calculations. Compared with FKM, the proposed method MCDFKM with $M = 1$ can reduce the computing time by a factor of 2.6 to 3.1 for the data sets generated with cluster centers evenly distributed over a hypercube. Experimental results show that the proposed approaches and FKM can obtain the same

clustering result. The proposed methods are used to reduce the computational complexity of conventional fuzzy *k*-means clustering. Therefore the Euclidean distance is used as the distortion measure. However, the proposed method can be extended to other distortion measure, such as Hamming distance.

REFERENCES

1. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Boston, MA, 1991.
2. Y. C. Liaw, J. Z. C. Lai, and W. Lo, "Image restoration of compressed image using classified vector quantization," *Pattern Recognition*, Vol. 35, 2002, pp. 181-192.
3. J. Foster, R. M. Gray, and M. O. Dunham, "Finite state vector quantization for waveform coding," *IEEE Transactions on Information Theory*, Vol. 31, 1985, pp. 348-359.
4. J. Z. C. Lai, Y. C. Liaw, and W. Lo, "Artifact reduction of JPEG coded images using mean-removed classified vector quantization," *Signal Processing*, Vol. 82, 2002, pp. 1375-1388.
5. S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, Academic Press, 2nd ed., New York, 2003.
6. U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining*, MIT Press, Boston, MA, 1996.
7. D. Liu and F. Kubala, "Online speaker clustering," in *Proceedings of IEEE Conference on Acoustic*, *Speech*, *and Signal Processing*, Vol. 1, 2004, pp. 333-336.
8. P. Hojen-Sorensen, N. de Freitas, and T. Fog, "On-line probabilistic classification with particle filters," in *Proceedings of IEEE Signal Processing Society Workshop*, Vol. 1, 2000, pp. 386-395.
9. M. Eirinaki and M. Vazirgiannis, "Web mining for web personalization," *ACM Transactions on Internet Technology*, Vol. 3, 2003, pp. 1-27.
10. I. Berget, B. H. Mevik, and T. Naæs, "New modifications and applications of fuzzy *c*-means methodology," *Computational Statistics and Data Analysis*, Vol. 52, 2008, pp. 2403-2418.
11. C. Döring, M. J. Lesot, and R. Kruse, "Data analysis with fuzzy clustering methods," *Computational Statistics and Data Analysis*, 2006, pp. 192-214.
12. J. L. Fan, W. Z. Zhen, and W. X. Xie, "Suppressed fuzzy *c*-means clustering algorithm," *Pattern Recognition Letters*, 2003, pp. 1607-1612.
13. T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient *k*-means clustering algorithm: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, pp. 881-892.
14. D. Pelleg and A. Moore, "Accelerating exact *k*-means algorithms with geometric reasoning," in *Proceedings of ACM SIGKDD International Conference on Knowledge and Data Mining*, 1999, pp. 277-281.
15. J. Z. C. Lai, T. J. Huang, and Y. C. Liaw, "A fast k-means clustering algorithm using cluster center displacement," *Pattern Recognition*, Vol. 42, 2009, pp. 2551-2556.
16. B. U. Shankar and N. Pal, "FFCM: An effective method for large data sets," in *Proceedings of the 3rd International Conference on Fuzzy*, *Logic*, *Neural Nets*, *and Soft Computing*, 1994, pp. 331-332.

17. R. L. Cannon, J. V. Dave, and J. C. Bezdek, "Efficient implementation of the fuzzy *c*-means clustering algorithm," *IEEE Transactions on PAMI*, Vol. 8, 1986, pp. 248-255.
18. F. Höppner, "Speeding up fuzzy *c*-means using a hierarchical data organization to control the precision of membership calculation," *Fuzzy Sets and Systems*, Vol. 128, 2002, pp. 365-376.
19. J. Z. C. Lai, Y. C. Liaw, and J. Liu, "A fast VQ codebook generation algorithm using codeword displacement," *Pattern Recognition*, Vol. 41, 2008, pp. 315-319.
20. http://www.ics.uci/~mlearn/MLRespository.html.

Chih-Tang Chang (張智堂) received his B.S. and M.S. degrees in Computer Science and Engineering from National Taiwan Ocean University, Taiwan, in 2001 and 2004, respectively. He is currently working toward the Ph.D. degree in the area of multimedia communication. His current interests are in image processing, optical communication, and multimedia.

Jim Z. C. Lai (賴榮滄) received his Ph.D. in Engineering from the University of California, Los Angeles, in 1986. He then worked as a research engineer at GM and a lead project engineer at Rockwell International. He joined the Department of Information Engineering and Computer Science in 1988 as an associate professsor. From 1994 to 2003, he was the full professor at the same department. Since 2004, he has been the full professor at the Department of Computer Science, National Taiwan Ocean University. His current interests are in image processing, multimedia, and network security.

Mu-Der Jeng (鄭慕德) received the B.S. and M.S. degrees in Electrical Engineering from the National Cheng Kung University, Tainan, Taiwan, R.O.C., in 1983 and 1985, respectively, and the Ph.D. degree in Computer and Systems Engineering from the Rensselaer Polytechnic Institute, Troy, NY, in 1992. Since August 1992, he has been with the National Taiwan Ocean University, Keelung, Taiwan, R.O.C., where he is currently a Full Professor at the Department of Electrical Engineering. His current research interests include Petri nets, discrete event systems, computer-integrated manufacturing, semiconductor factory automation, embedded systems, and supervisory control of underwater robots.