Pattern Recognition 43 (2010) 1954–1963

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Fast global *k*-means clustering using cluster membership and inequality

Jim Z.C. Lai [a], Tsung-Jen Huang [a,b,*]

[a] Department of Computer Science and Engineering, National Taiwan Ocean University, Keelung 202, Taiwan
[b] Department of Information and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu 310, Taiwan

ARTICLE INFO

Article history:
Received 4 December 2008
Received in revised form
10 November 2009
Accepted 22 November 2009

Keywords:
Global *k*-means clustering
Nearest-neighbor search
Knowledge discovery

ABSTRACT

In this paper, we present a fast global *k*-means clustering algorithm by making use of the cluster membership and geometrical information of a data point. This algorithm is referred to as MFGKM. The algorithm uses a set of inequalities developed in this paper to determine a starting point for the *j*th cluster center of global *k*-means clustering. Adopting multiple cluster center selection (MCS) for MFGKM, we also develop another clustering algorithm called MFGKM+MCS. MCS determines more than one starting point for each step of cluster split; while the available fast and modified global *k*-means clustering algorithms select one starting point for each cluster split. Our proposed method MFGKM can obtain the least distortion; while MFGKM+MCS may give the least computing time. Compared to the modified global *k*-means clustering algorithm, our method MFGKM can reduce the computing time and number of distance calculations by a factor of 3.78–5.55 and 21.13–31.41, respectively, with the average distortion reduction of 5,487 for the Statlog data set. Compared to the fast global *k*-means clustering algorithm, our method MFGKM+MCS can reduce the computing time by a factor of 5.78–8.70 with the average reduction of distortion of 30,564 using the same data set. The performances of our proposed methods are more remarkable when a data set with higher dimension is divided into more clusters.

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 a cost function, *k*-means clustering is perhaps the most widely used and studied [10]. The *k*-means clustering algorithm, which is also called the generalized Lloyd algorithm (GLA), is a special case of the generalized hard clustering scheme, when point representatives are adopted and the squared Euclidean distances are used to measure the distortion (dissimilarity) between a vector **X** and its cluster representative (cluster center) **C**.

The *k*-means clustering algorithm performs iteratively the partition step and new cluster center generation step until convergence. It is noted that the *k*-means clustering algorithm faces the local minimum problem. That is, the clustering results guarantee local minimum distortion only. To solve this problem, the simulated annealing method may be used [11]. This method usually needs a very large amount of additional computations and obtains a small distortion improvement. The global *k*-means clustering algorithm (GKM), which is an incremental algorithm, is proposed to solve the local minimum problem too [12]. This algorithm uses each data point as a starting one for the *j*th cluster center, where $1 \le j \le k$ and k is the number of clusters. GKM may lead to a near global solution, although it needs a very huge amount of computing time. In Ref. [12], a fast global *k*-means clustering algorithm (FGKM) is also presented to reduce the computational complexity of GKM. A modified global *k*-means clustering algorithm (MGKM) is proposed in Ref. [13] to reduce the computational load of GKM also. MGKM minimizes an auxiliary function to choose a staring point for the *j*th cluster center. Compared to FGKM, MGMK can obtain a slightly better result with the longer computing time. Both GKM and MGKM use GLA to solve the partition problem. MGKM needs $O(N^2)$ memory space to store distances between all data points, where N is the number of data objects. For a large data set, it is infeasible to store these distances between all data points. It is noted that both GKM and MGKM are not targeted for a large data set.

The computing time of MGKM or GKM can be reduced through decreasing the computational complexity of GLA. Some fast *k*-means clustering algorithms are available [10,14]. Kanungo et al. [10] developed a filtering algorithm on a kd-tree to speed up the generation of new cluster centers. The idea of sorting data points in a kd-tree for *k*-means clustering was also used by Pelleg and Moore [15]. The kd-tree based algorithms are not suitable for a data set with high dimension, since their computational complexities grow exponentially with the data dimension. Lai and Liaw developed a modified filtering algorithm to reduce the

* Corresponding author at: Department of Information and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu 310, Taiwan.
E-mail addresses: ph0427@gmail.com, D97570003@ntou.edu.tw (T.-J. Huang).

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.11.021

computing time of a kd-tree based algorithm [16]. Lai et al. [14] used the information of cluster activity to speed up the process of cluster center generation. In Ref. [14], the cluster centers were classified into static and active groups. In many cases, the nearest cluster center of a data point was determined from the active group in the partition step.

The geometrical information of data points can be used to reduce the computational complexity of MGKM. The fast search algorithms used to reduce the search complexity of the partition step of GLA [17–19] can also be adopted to speed up the partition process of global k-means clustering. The mean-distance-ordered partial search (MPS) algorithm [17] eliminated impossible cluster centers using the difference between mean values of a data point and cluster center. Lai and Liaw [18] presented a fast-searching algorithm using projection and inequality (FAUPI) to reject unlikely cluster centers for a data point. Lai et al. [19] also developed a fast k-nearest-neighbor search (FKNNS) algorithm using projection and triangular inequality to reject unlikely candidates in the process of finding k nearest neighbors for a query point.

In this paper, we will present an algorithm to reduce the computing time of global k-means clustering. The algorithm first uses a set of inequalities developed in this paper to determine a staring point for the jth cluster center of global k-means clustering and then modifies the method developed in Ref. [14] to speed up the corresponding partition process. This paper is organized as follows. Section 2 describes the fast global and modified global k-means clustering algorithms. Section 3 presents the algorithms developed in this paper. Experimental results are presented in Section 4 and concluding remarks are given in Section 5.

2. Fast and modified global k-means clustering algorithms

The k-means clustering algorithm partitions data points into k clusters S_j (j=1,2,…,k) associated with representatives (cluster centers) $\mathbf{C}_j$. Denote the set of data points as $S=\{\mathbf{X}_m\}$, where m=1,2,…,N and N is the number of data points in the set S. Let $\mathrm{d}(\mathbf{X},\mathbf{Y})$ be the distortion between any two vectors $\mathbf{X}$ and $\mathbf{Y}$. In this paper, $\mathrm{d}(\mathbf{X},\mathbf{Y})$ is defined as the squared Euclidean distance between $\mathbf{X}$ and $\mathbf{Y}$.

Let $\mathbf{C}_{nm}$ be the nearest cluster center of $\mathbf{X}_m$ and $d_m=\mathrm{d}(\mathbf{X}_m, \mathbf{C}_{nm})$. The goal of GLA is to find a set of cluster centers $SC=\{\mathbf{C}_l\}$ such that the distortion J defined below is minimized, where l=1 to k and k is the number of clusters.

$$J=\sum_{m=1}^{N} d_m \tag{1}$$

The major process of GLA (k-means clustering) is mapping a given set of representative vectors into an improved one through partitioning data points. It begins with an initial set of cluster centers and repeats this mapping process until a stopping criterion is satisfied. The Lloyd iteration for the k-means clustering is given as follows [1].

(1) Given a set of cluster centers $SC_p=\{\mathbf{C}_j\}$, find the partition of S; that is S is divided into k clusters S_l, where l=1, 2,…,k and $S_l=\{\mathbf{X}|\mathrm{d}(\mathbf{X}, \mathbf{C}_l)\le \mathrm{d}(\mathbf{X}, \mathbf{C}_j)$ for all $l\neq j\}$.
(2) Compute the centroid for each cluster to obtain a new set of cluster representatives SC_{p+1}.

The GLA is briefly described as follows.

(1) Begin with an initial set of cluster centers SC_0. Set p=0.
(2) Given the set of cluster centers SC_p, perform the Lloyd iteration to generate the improved set of cluster representatives SC_{p+1}.
(3) Compute the distortion J for SC_{p+1}. If it is changed by a small enough amount since the last iteration, then stop. Otherwise set $p+1\to p$ and go to step (2).

2.1. Fast global k-means clustering algorithm

For the fast global k-means clustering, a starting point is determined at each step of cluster split. Let the set S be partitioned into $(j-1)$ clusters S_l (l=1,2,…,$j-1$) with centers $\mathbf{C}_l$. Denote this set of cluster centers as $SC(j-1)=\{\mathbf{C}_1,\ldots,\mathbf{C}_{j-1}\}$. To determine a starting point for the jth cluster center, a value a_m is calculated by the following equation.

$$a_m=\sum_{i=1}^{N}\max\{0, d_i-\mathrm{d}(\mathbf{X}_i,\mathbf{X}_m)\}, m=1,2,\ldots,N \tag{2}$$

A data point $\mathbf{X}_{j'}\in S$ with the largest value of $a_{j'}$ is then selected as a staring point for the jth cluster center. After the staring point $\mathbf{X}_{j'}$ is found, FGKM (fast global k-means clustering algorithm) uses $\{\mathbf{C}_1,\ldots,\mathbf{C}_{j-1},\mathbf{X}_{j'}\}$ as the initial set of cluster centers to partition a data set. The resulting set of cluster centers is denoted as $SC^*(j)=\{\mathbf{C}_1^*,\ldots,\mathbf{C}_j^*\}$. Now we can present the fast global k-means clustering algorithm (FGKM) as follows:

Fast global k-means clustering algorithm

(1) Compute $SC(1)=\{\mathbf{C}_1\}$ from the set S, where $\mathbf{C}_1=\frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_i$ and N is the number of data points. Set j=2.
(2) Find the data point $\mathbf{X}_{j'}\in S$ with the largest value of $a_{j'}$. Let $SC'(j)=\{\mathbf{C}_1,\ldots,\mathbf{C}_{j-1},\mathbf{X}_{j'}\}$.
(3) Use $SC'(j)$ as the set of initial cluster centers and apply GLA to find $SC^*(j)=\{\mathbf{C}_1^*,\ldots,\mathbf{C}_j^*\}$, which partitions S into j clusters. Set $SC(j)=\{\mathbf{C}_1,\ldots,\mathbf{C}_j\}=\{\mathbf{C}_1^*,\ldots,\mathbf{C}_j^*\}$.
(4) Set $j=j+1$. If $j>k$, stop; otherwise go to step (2).

To add a starting point for the jth cluster center, FGKM needs $O(N^2)$ distance calculations. After adding a new cluster center, using GLA to calculate the new cluster centers for FGKM requires $O(Nkt)$ distance calculations, where t is the number of iterations performed by GLA. To generate a set of k cluster centers, FGKM needs $O((N^2+Nkt)k)=O(N^2k+N\ k^2t)$ distance calculations. FGKM can store the pairwise data distances and thus completely avoid distance calculations when selecting a newly added cluster center.

2.2. Modified global k-means clustering algorithm

The difference between FGKM and MGKM is that they use different methods to determine a starting point for the jth cluster center. Given this set of $(j-1)$ cluster centers $SC(j-1)=\{\mathbf{C}_1,\ldots,\mathbf{C}_{j-1}\}$, denote d_m as the squared distance between $\mathbf{X}_m$ and its closest cluster center for each $\mathbf{X}_m\in S$. Let the set $R(\mathbf{X}_m)=\{\mathbf{X}_i|\mathrm{d}(\mathbf{X}_i,\mathbf{X}_m)<d_i$ for all $\mathbf{X}_i\in S\}$. Denote $\mathbf{Y}_m$ as the center of the set $R(\mathbf{X}_m)$ and define $F(\mathbf{Y}_m)$ as

$$F(\mathbf{Y}_m)=\sum_{i=1}^{N}\min\{d_i, \mathrm{d}(\mathbf{X}_i,\mathbf{Y}_m)\}, m=1,2,\ldots,N \tag{3}$$

Let $\mathbf{Y}_{j'}$ be the center of $R(\mathbf{Y}_{j'})$ such that $F(\mathbf{Y}_{j'})\le F(\mathbf{Y}_m)$, where $j'\neq m$, $1\le j'$, $m\le N$, and $R(\mathbf{Y}_{j'})=\{\mathbf{X}_i|\mathrm{d}(\mathbf{X}_i,\mathbf{Y}_{j'})<d_i$ for all $\mathbf{X}_i\in S\}$. After $\mathbf{Y}_{j'}$ is determined, we can recalculate the set $R(\mathbf{Y}_{j'})$. From the set $R(\mathbf{Y}_{j'})$, we can determine its center $\mathbf{Y}_{j'}$ again. These procedures will be

repeated until $\mathbf{Y}_{j'}$ is converged. Now, we may present the initial point selection algorithm for MGKM as follows:

Initial point selection algorithm

(1) For each data point $\mathbf{X}_m \in S$, find the set $R(\mathbf{X}_m)=\{\mathbf{X}_i | \mathrm{d}(\mathbf{X}_i, \mathbf{X}_m) < d_i$ for all $\mathbf{X}_i \in S\}$. Compute $F(\mathbf{Y}_m)$ using Eq. (3), where $\mathbf{Y}_m$ is the center of $R(\mathbf{X}_m)$.
(2) Determine $\mathbf{Y}_{j'}$ such that $F(\mathbf{Y}_{j'}) \le F(\mathbf{Y}_m)$, where $j' \ne m$ and $1 \le j'$, $m \le N$.
(3) Find the set $R(\mathbf{Y}_{j'}) = \{\mathbf{X}_i | \mathrm{d}(\mathbf{X}_i, \mathbf{Y}_{j'}) < d_i$ for all $\mathbf{X}_i \in S\}$.
(4) Determine $\mathbf{Y}_{j'}$ which is the center of $R(\mathbf{Y}_{j'})$.
(5) Repeat steps (3) and (4) until $\mathbf{Y}_{j'}$ is converged.

At this stage, we would like to present the modified global k-means clustering algorithm (MGKM) below:

Modified global k-means clustering algorithm

(1) Compute $SC(1)=\{\mathbf{C}_1\}$ from the set S, where $\mathbf{C}_1 = \frac{1}{N}\sum_{i=1}^{N} \mathbf{X}_i$ and N is the number of data points. Calculate the distances between all data points and store these distances. Set $j=2$.
(2) Use the initial point selection algorithm to determine $\mathbf{Y}_{j'}$. Let $SC'(j) = \{\mathbf{C}_1, \ldots, \mathbf{C}_{j-1}, \mathbf{Y}_{j'}\}$.
(3) Use $SC'(j)$ as the set of initial cluster centers and apply GLA to find $SC^*(j) = \{\mathbf{C}_1^*, \ldots, \mathbf{C}_j^*\}$, which partitions S into j clusters, Set $SC(j) = \{\mathbf{C}_1, \ldots, \mathbf{C}_j\} = \{\mathbf{C}_1^*, \ldots, \mathbf{C}_j^*\}$.
(4) Set $j=j+1$. If $j > k$, stop; otherwise go to step (2).

MGKM and FGKM have the same computational complexity, in terms of the number of distance calculations, when the pairwise distances are not stored for FGKM. That is, MGKM needs $\mathrm{O}(N^2k+Nk^2t)$ distance calculations to partition a data set into k clusters.

3. Proposed algorithms

3.1. Modified fast global k-means clustering algorithm

In this section, we will develop a set of inequalities to speed up global k-means clustering. Let the set $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d\}$ consist of d eigenvectors obtained from a set of data points $\{\mathbf{X}_m\}$ using KLT [20,21], where d is the data dimension. Without loss of generality, we assume that their corresponding eigenvalues λ_i ($i=1,2,\ldots,d$) are arranged in the descending order. An eigenvector with larger eigenvalue implies that the distribution of projections of data points on the corresponding axis is more scattered. That is, using the differences of projections of data points on an axis with larger eigenvalue will have the better chance to reject unlikely candidates for the nearest neighbor of a data point. The number of projections used for k-nearest-neighbor search should be dependent on data distribution. However, three projection values are used for the k-nearest-neighbor search [18,19,22]. As shown in Ref. [22], the performance of rejecting unlikely candidates for a vector's nearest neighbor is not significantly improved when more than three axes are used. The number of eigenvectors used for our method should be determined such that their corresponding values should be larger than a threshold. Denote $\mathbf{v}_1$ to $\mathbf{v}_{n_a}$ as the n_a eigenvectors corresponding to the n_a largest eigenvalues. Using power method, we need $\mathrm{O}(Nd^2+d^2tn_a) \approx \mathrm{O}(Nd^2)$ multiplications and $\mathrm{O}(Nd^2+d^2tn_a) \approx \mathrm{O}(Nd^2)$ additions to obtain n_a largest eigenvectors [23], where d is the data dimension and $t \ll N$ is the number of iterations to obtain eigenvectors with enough accuracy. Let x_{mi} be the projection value of a data point $\mathbf{X}_m$ on the ith axis. That is, x_{mi} is the inner product of $\mathbf{X}_m$ and $\mathbf{v}_i$, and can be calculated as follows:

$$x_{mi} = \langle \mathbf{X}_m, \mathbf{v}_i \rangle \tag{4}$$

Similarly, denote q_i as the projection value of a candidate point $\mathbf{Q}$ on the ith axis. To speed up the searching process, all data points are sorted in the ascending order of their projections on the first axis. If

$$|x_{mi}-q_i|^2 \ge d_m \tag{5}$$

then $\|\mathbf{X}_m-\mathbf{Q}\|^2 = \sum_{i=1}^{d} |x_{mi}-q_i|^2 \ge d_m$, where d_m is the squared Euclidean distance between the data point $\mathbf{X}_m$ and its closest cluster center. That is, if expression (5) is satisfied, then point $\mathbf{Q}$ cannot be the nearest neighbor of a data point $\mathbf{X}_m$.

Let $\mathbf{q} = \sum_{i=1}^{n_a} q_i\mathbf{v}_i$, $\mathbf{x}_m = \sum_{i=1}^{n_a} x_{mi}\mathbf{v}_i$, $\mathbf{s}_\mathrm{q}=\mathbf{Q}-\mathbf{q}$, and $\mathbf{s}_{\mathrm{xm}}=\mathbf{X}_m-\mathbf{x}_m$. From the definitions of $\mathbf{q}$, $\mathbf{x}_m$, $\mathbf{s}_\mathrm{q}$, and $\mathbf{s}_{\mathrm{xm}}$, we may conclude that $\mathbf{x}_m \perp \mathbf{s}_{\mathrm{xm}}$, $\mathbf{x}_m \perp \mathbf{s}_\mathrm{q}$, $\mathbf{q} \perp \mathbf{s}_{\mathrm{xm}}$, and $\mathbf{q} \perp \mathbf{s}_\mathrm{q}$. At this stage, the following equation may be obtained accordingly.

$$\begin{aligned}\|\mathbf{X}_m-\mathbf{Q}\|^2 &= \langle [(\mathbf{x}_m-\mathbf{q})+(\mathbf{s}_{xm}-\mathbf{s}_q)],\ [(\mathbf{x}_m-\mathbf{q})+(\mathbf{s}_{xm}-\mathbf{s}_q)] \rangle \\ &= \|\mathbf{x}_m-\mathbf{q}\|^2 + \|\mathbf{s}_{xm}-\mathbf{s}_q^2\end{aligned} \tag{6a}$$

From Eq. (6a), we can easily show that

$$\|\mathbf{X}_m-\mathbf{Q}\|^2 \ge \|\mathbf{x}_m-\mathbf{q}\|^2 + (\|\mathbf{s}_{\mathrm{xm}}\|-\|\mathbf{s}_\mathrm{q}\|)^2 \tag{6b}$$

where

$\|\mathbf{s}_{\mathrm{xm}}\| = (\|\mathbf{X}_m\|^2-\|\mathbf{x}_\mathrm{m}\|^2)^{0.5}$ and $\|\mathbf{s}_\mathrm{q}\| = (\|\mathbf{Q}\|^2-\|\mathbf{q}\|^2)^{0.5}$. If

$$\|\mathbf{x}_m-\mathbf{q}\|^2 + (\|\mathbf{s}_{\mathrm{xm}}\|-\|\mathbf{s}_\mathrm{q}\|)^2 \ge d_m \tag{6c}$$

then $\|\mathbf{X}_m-\mathbf{Q}\|^2 \ge d_m$. Suppose that the set S is partitioned into $(j-1)$ clusters S_l ($l=1,2,\ldots j-1$) with centers $\mathbf{C}_l$. Let $r_{max}=\max\{d_1, d_2, \ldots, d_N\}$. If S is to be partitioned into j clusters using $SC' = \{\mathbf{C}_1, \ldots, \mathbf{C}_{j-1}, \mathbf{Q}\}$ as the set of cluster centers, where $\mathbf{Q}$ is a candidate point, we should determine $R(\mathbf{Q})$, which is a set of data points whose nearest cluster center is $\mathbf{Q}$. Using inequality (5), we may conclude that if the candidate point $\mathbf{Q}$ cannot satisfy the following condition, then we may conclude that it cannot be the nearest cluster center of a data point $\mathbf{X}_m$ in the data set. That is, $\mathbf{X}_m$ can be rejected directly in the process of constructing $R(\mathbf{Q})$.

$$|x_{mi}-q_i|^2 \ge d_m, i = 1 \text{ to } n_a \tag{7a}$$

where q_i and x_{mi} are the projection values of $\mathbf{Q}$ and $\mathbf{X}_m$, respectively, on the ith axis. Similarly, if expression (6c) is satisfied, then $\mathbf{X}_m$ can also be rejected directly in the process of constructing $R(\mathbf{Q})$. If the candidate point $\mathbf{Q}$ is the nearest neighbor of a data point $\mathbf{X}_m$, then their projection values on the first axis may be very close. For any data point $\mathbf{X}$ in the data set S, if expression (7b) is satisfied then $\mathbf{X}$ is impossible to be in $R(\mathbf{Q})$ also.

$$|x_{mi}-q_i|^2 \ge r_{max}, i = 1 \text{ to } n_a \tag{7b}$$

where x_i is the projection value of $\mathbf{X}$ on the ith axis. In the process of constructing $R(\mathbf{Q})$, a data point, whose projection on the first axis is close to q_1, is first selected as the starting point. Now, we can present the candidate set construction algorithm to determine $R(\mathbf{Q})$ and $F(\mathbf{Q})$, defined by Eq. (3), as follows:

Candidate set construction algorithm

(1) Input $\{\mathbf{C}_1, \ldots, \mathbf{C}_{j-1}\}$ and the value of n_a, candidate data point $\mathbf{X}_m$, the distortion J, r_{max}, repeat times T, and the distances d_j between $\mathbf{X}_j$ ($1 \le j \le N$) and their corresponding closest cluster centers, where $j=1$ to N. Set $\mathbf{Q}=\mathbf{X}_m$, $q_i=x_{mi}$ ($i=1$ to n_a), $diff=0$

and $R(\mathbf{Q})=\emptyset$ where $\emptyset$ is the empty set. Let $prev\_diff=-\infty$, $t=1$, $l=m$, $p=0$, $LB=1$ and $UB=N$.

(2) If $(l+p)>UB$ or the data point $\mathbf{X}_{l+p}$ is deleted, go to step (3). Otherwise, let $r=d_{l+p}$ and go to step (2.1), where r is the squared Euclidean distances between the data point $\mathbf{X}_{l+p}$ and its closest cluster center.
  (2.1) Compute $D_1=|x_{(l+p)1}-q_1|^2$. If $D_1\geq r_{max}$, then delete all data points from $\mathbf{X}_{l+p}$ to $\mathbf{X}_N$, set $UB=l+p$, and go to step (3). Otherwise set $n=2$.
  (2.2) Compute $D_n=D_{n-1}+|x_{(l+p)n}-q_n|^2$. If $D_n\geq r$, then delete $\mathbf{X}_{l+p}$ and go to step (3).
  (2.3) Set $n=n+1$. Repeat step (2.2) until $n>n_a$.
  (2.4) Calculate $\|\mathbf{s}_q\|$. If $\|\mathbf{x}_{(l+p)}-\mathbf{q}\|^2+(\|\mathbf{s}_{(l+p)}\|-\|\mathbf{s}_q\|)^2\geq r$, then delete $\mathbf{X}_{l+p}$ and go to step (3).
  (2.5) Calculate $r'=\mathrm{d}(\mathbf{X}_{l+p},\mathbf{Q})$. If $r'<r$, then put $\mathbf{X}_{l+p}$ in $R(\mathbf{Q})$, update $diff=diff+(r-r')$ and go to step (3).

(3) If $(l-p)<LB$ or the data point $\mathbf{X}_{l-p}$ is deleted, go to step (4). Otherwise, let $r=d_{l-p}$ and go to step (3.1).
  (3.1) Compute $D_1=|x_{(l-p)1}-q_1|^2$. If $D_1\geq r_{max}$, then eliminate all data points from $\mathbf{X}_{l-p}$ to $\mathbf{X}_1$, set $LB=l-p$, and go to step (4). Otherwise set $n=2$.
  (3.2) Compute $D_n=D_{n-1}+|x_{(l-p)n}-q_n|^2$. If $D_n\geq r$, then delete $\mathbf{X}_{l+p}$ and go to step (4).
  (3.3) Set $n=n+1$. Repeat step (3.2) until $n>n_a$.
  (3.4) Calculate $\|\mathbf{s}_q\|$. If $\|\mathbf{x}_{(l-p)}-\mathbf{q}\|^2+(\|\mathbf{s}_{(l-p)}\|-\|\mathbf{s}_q\|)^2\geq r$, then delete $\mathbf{X}_{l-p}$ and go to step (4).
  (3.5) Calculate $r'=\mathrm{d}(\mathbf{X}_{l-p},\mathbf{Q})$. If $r'<r$, then put $\mathbf{X}_{l-p}$ in $R(\mathbf{Q})$ and update $diff=diff+(r-r')$.

(4) Set $p=p+1$. If $((l+p)>UB$ and $(l-p)<LB)$, go to step (5). Otherwise, go to step (2).

(5) If $diff<prev\_diff$, stop the process and go to step (7). Otherwise, Calculate the center of $R(\mathbf{Q})$, which is denoted as $\mathbf{Y}$.

(6) Set $t=t+1$. If $t<T$, set $prev\_diff=diff$, $diff=0$, $LB=1$, $UB=N$, $p=0$, $R(\mathbf{Q})=\emptyset$, update $\mathbf{Q}=\mathbf{Y}$, and calculate q_i $(i=1, 2,, n_a)$. Find a data point $\mathbf{X}_l$ whose projection on the first axis is closest to the projection of $\mathbf{Y}$ on the first axis and go to step (2).

(7) Calculate $F(\mathbf{Y})$, where $F(\mathbf{Y})=J-diff$, and output the cell center $\mathbf{Y}$ as well as $F(\mathbf{Y})$.

Step (1) of the candidate set construction algorithm performs the initialization process. Step (2) of this algorithm determines data points whose nearest neighbor is $\mathbf{Q}$ and the corresponding projections on the first axis are larger than q_1. Similarly, step (3) finds all data points whose nearest neighbor is $\mathbf{Q}$ and the corresponding projections on the first axis are less than q_1. That is, steps (2) to (4) determine all data points whose nearest neighbor is $\mathbf{Q}$. Steps (5) and (6) calculate the center of data points with $\mathbf{Q}$ as their nearest neighbor. At step (5) of the candidate set construction algorithm, if $(diff<prev\_diff)$ is satisfied, then a smaller value of $F(\mathbf{Y})$ is obtained in the previous iteration and this algorithm is terminated. Finally step (7) outputs the cell center Y and its corresponding value $F(\mathbf{Y})$.

It is noted that the number of axes n_a used for the candidate set construction algorithm is dependent on data distribution in general. Let E_n be a ratio defined by the following equation:

$$\mathrm{E}_n=\frac{\sum_{i=1}^{n}\lambda_i^2}{\sum_{i=1}^{d}\lambda_i^2} \tag{8}$$

where d is the data dimension, n is the number of axes, and λ_i is the ith largest eigenvalue. n_a may be selected by the following expression:

$$n_a=\underset{n}{\mathrm{argmin}}(\mathrm{E}_n>th) \tag{9}$$

where th is a threshold and $n_a<d$. From our experiences, $th=0.9999$ may be adopted.

At this stage, we would like to present the cluster center selection algorithm to determine the initial cluster center for the jth cluster.

Cluster center selection algorithm

(1) For each data point $\mathbf{X}_m\in S$, use the candidate set construction algorithm to determine $\mathbf{Y}_m$ and $F(\mathbf{Y}_m)$.
(2) Determine $\mathbf{Y}_{j'}$ such that $F(\mathbf{Y}_{j'})\leq F(\mathbf{Y}_m)$ for $j'\neq m$ and $1\leq m\leq N$.

After determining the initial center for the jth cluster, our method uses CGAUCD (codebook generation algorithm using codeword displacement) presented in Ref. [14] to obtain the set of cluster centers $SC(j)=\{\mathbf{C}_1,\ldots,\mathbf{C}_j\}$. It is noted that compared to GLA, CGAUCD has less computing time. GLA as well as CGAUCD will obtain the same set of clusters, if the same set of initial cluster centers is used [14]. Now, we would like to present the proposed fast global k-means clustering algorithm using cluster membership and inequality, which is referred to as MFGKM, as follows:

Modified fast global k-means clustering algorithm using cluster membership and inequality

(1) Determine all data points' projection values on the first to the n_ath axes and arrange all these data points in the ascending order of projections on the first axis.
(2) Compute $SC(1)=\{\mathbf{C}_1\}$ from the data set S, where $\mathbf{C}_1=\frac{1}{N}\sum_{i=1}^{N}\mathbf{X}_i$ and N is the number of data points. Determine r_{max} and set $j=2$.
(3) Use the cluster center selection algorithm to determine $\mathbf{Y}_{j'}$, which is the initial cluster center for the jth cluster. Let $SC'(j)=\{\mathbf{C}_1,\ldots,\mathbf{C}_{j-1},\mathbf{Y}_{j'}\}$.
(4) Use $SC'(j)$ as the set of initial cluster centers and apply CGAUCD to find $SC^*(j)=\{\mathbf{C}_1^*,\ldots,\mathbf{C}_j^*\}$, which partitions S into j clusters, and the value of r_{max}. Set $SC(j)=\{\mathbf{C}_1,\ldots,\mathbf{C}_j\}=\{\mathbf{C}_1^*,\ldots,\mathbf{C}_j^*\}$.
(5) Set $j=j+1$. If $j>k$, stop; otherwise go to step (3).

Our algorithm uses power method [23] to find n_a largest eigenvectors from a set of N data points. This requires $O(Nd^2)$ multiplications and $O(Nd^2)$ additions, where d is the data dimension. One distance calculation needs d multiplications and $(2d-1)$ additions. Thus, the above process requires about $O(Nd)$ distance calculations. Our method also uses the cluster center selection algorithm to determine the initial cluster center for the jth cluster. This process needs $O(NN_1)$ distance calculations, where N_1 is the number of distance calculations executed by the candidate set construction algorithm and $N_1\ll N$. To add a cluster to the existing set, we need $O(NN_1+Nk_1t)$ distance calculations, where t is the number of iterations performed by CGAUCD, k is the number of clusters, and k_1 is the average number of distance calculations performed by CGAUCD for each data point and stage of iteration. It is noted that Nk_1t is the computational complexity, in terms of the number of distance calculations, of CGAUCD. To generate a set of k cluster centers using our method, we require $O(NN_1k+Nk_1kt+Nd)$ distance calculations, where $N_1\ll N$ and $k1\ll k$. Since the computational complexities, in terms of the number of distance calculations, of FGKM and MGKM are $O(N^2k+Nk^2t)$, we may conclude that our method has less computational complexity.

3.2. Multiple cluster center selection algorithm

We will also present MFGKM with multiple cluster selection (MCS) technique, referred to as MFGKM+MCS, in this paper. This

algorithm increases s cluster centers in each step. MCS uses the same process as that of cluster center selection algorithm to determine s starting points. Since the computational complexity of selecting s starting points is the same as that of finding only one, we may expect that MFGKM+MCS will reduce the computing time of MFGKM significantly. Let $SC(j)=\{\mathbf{C}_1,\mathbf{C}_2,\ldots,\mathbf{C}_j\}$ consist of j cluster centers. To generate $(j+s)$ cluster centers using MFGKM+MCS, we will modify the cluster center selection algorithm described previously to select s starting points $\mathbf{Y}_{j'}$, where $j'=1, 2,\ldots, s$. We determine $\mathbf{Y}_{j'}$ such that their $F(\mathbf{Y}_{j'})$ are minimum and $d(\mathbf{Y}_{m'},\mathbf{Y}_{n'})>thd$, where thd is a threshold, $m'\neq n'$, $1\leq m', n'\leq s$, and $j'=1$ to s. In this paper, we set thd as the average distortion per data point. That is $thd=J/N$, where N is the number of data points and J is the total distortion with j cluster centers. It is noted that the value of thd can be calculated by CGAUCD. For our method, $\mathbf{Y}_{j'}$ are sorted in the ascending order of $F(\mathbf{Y}_{j'})$. The multiple cluster center selection (MCS) algorithm is briefly described below:

Multiple cluster center selection algorithm

(1) Input the values of s and n_a, the number of cluster centers k, and the set of current clusters $SC(j)$.
(2) Set $s=\min(s, k-j)$.
(3) For each data point $\mathbf{X}_m\in S$, use the candidate set construction algorithm to determine $\mathbf{Y}_m$ and $F(\mathbf{Y}_m)$.
(4) Determine $\mathbf{Y}_{j'}$ such that their $F(\mathbf{Y}_{j'})$ are minimum and $d(\mathbf{Y}_{m'},\mathbf{Y}_{n'})>thd$, where thd is the average distortion per data point, $m'\neq n'$, $1\leq m', n'\leq s$, and $j'=1$ to s. Arrange $\mathbf{Y}_{j'}$ in the ascending order of $F(\mathbf{Y}_{j'})$.

The computational complexity, in terms of the number of distance calculations, of MFGKM+MCS is $O(Nd+NN_1k/s+Nk_1kt/s)$, where $O(Nd)$ distance calculations are used to perform PCA (principal component analysis) to determine n_a largest eigenvectors, t is the number of iterations performed by CGAUCD, k_1 is the average number of distance calculations performed by CGAUCD for each data point and stage of iteration, s is the number of clusters being increased at each step of cluster split, and k is the number of clusters.

4. Experimental results

To evaluate the performance of the proposed algorithm, one synthetic data set and six real data sets have been used. In the first example, three data sets with $d=16$ are generated from three images: "Lena+Airplane," "Peppers," and "House." The block size for each data point is 4×4 in the first example. In the second example, the set of data points having 16,384 data points with dimension 16 is obtained from the real image: "Lena." This data set consists of 16,384 image blocks of 4×4 pixels. In example 3, the data set having 49,152 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 this example. In example 4, the Statlog data set consists of 6,435 data points with $d=36$ [24]. In example 5, several data sets with size 10,000 and dimensions from 8 to 40 are obtained from the Gauss Markov sequence [1] with $\sigma=1.0$, $\mu=0$, 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 these examples, the proposed algorithms MFGKM and MFGKM+MCS are compared to FGKM and MGKM in terms of the computing time, distortion, and number of distance calculations. To compare the performances of these methods, the Normalized Distortion is defined for an algorithm with distortion J.

$$\text{Normalized distortion}=\frac{J}{J_{\text{FGKM}}}$$

where J_{FGKM} is the distortion of FGKM. All computing is performed on an Intel Core2 Quad 2.40 GHz PC with 2 GB of memory. All programs are implemented as console applications of Microsoft Visual Studio 2005 and are executed under Window XP Professional SP3.

4.1. Example 1: three data sets obtained from images: "House," "Peppers," and "Lena+Airplane."

In this example, three data sets with $d=16$ are generated from real images: "House," "Peppers," and "Lena+Airplane." The first two data sets have 16,384 data points; while the third one is with $N=32{,}768$, where N is the number of data points. These data sets are used to test how the number of projection axes affects the computing time of our proposed method MFGKM. Fig. 1 shows the computing time of MFGKM with various values of E_n defined in Eq. (8). From Fig. 1, we may find that the decrease of computing time for MFGKM is not significant when E_n is larger than 0.9999. In the following examples, $E_n=0.9999$ is used for our proposed methods.

4.2. Example 2: a data set generated from a real image: "Lena."

In the second example, a data set with $d=16$ and $N=16{,}384$ is generated from a real image: "Lena." This data set is a small one and is used for testing the performances of FGKM, MGKM, MFGKM and MFGKM+MCS. Using Eq. (9), we can determine the

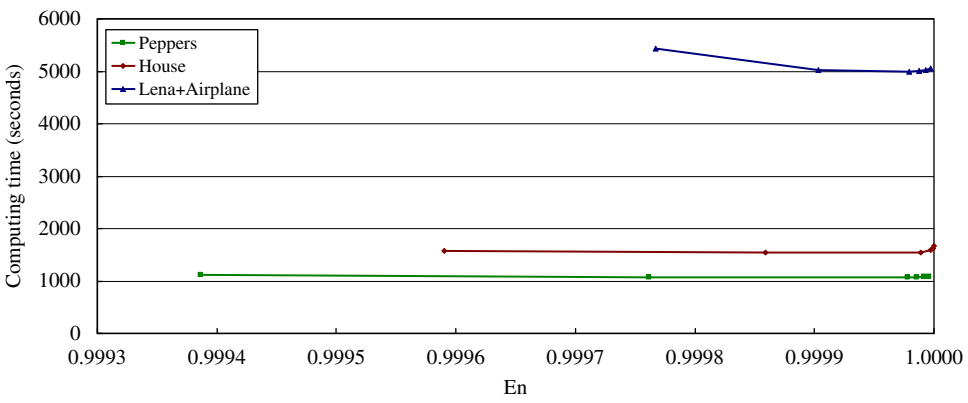

Fig. 1. The computing time (in seconds) of MFGKM ($T=2$, $k=256$) using three sets of real images: "House," "Peppers," and "Lena+Airplane."

Table 1
The computing time (in seconds) using a real image: "Lena".

Method	k		
	64	128	256
FGKM	495.53	998.03	2008.95
MGKM	1023.45	2028.25	4032.03
MFGKM (T=1)	180.83	310.27	526.33
MFGKM (T=2)	349.38	569.66	946.77
MFGKM+MCS (T=1, s=2)	107.17	186.95	309.38
MFGKM+MCS (T=1, s=4)	59.09	99.69	168.17
MFGKM+MCS (T=2, s=2)	209.19	348.67	569.75
MFGKM+MCS (T=2, s=4)	112.63	182.38	305.06

Table 2
The number of distance calculations using a real image: "Lena".

Method	k		
	64	128	256
FGKM	1.7732E+10	3.6295E+10	7.3957E+10
MGKM	3.4543E+10	7.0336E+10	1.4233E+11
MFGKM (T=1)	9.2654E+08	1.5319E+09	2.6939E+09
MFGKM (T=2)	1.5556E+09	2.4556E+09	4.0275E+09
MFGKM+MCS (T=1, s=2)	5.1309E+08	8.8784E+08	1.5810E+09
MFGKM+MCS (T=1, s=4)	3.4799E+08	5.7694E+08	1.0354E+09
MFGKM+MCS (T=2, s=2)	9.1925E+08	1.4729E+09	2.4461E+09
MFGKM+MCS (T=2, s=4)	5.5735E+08	8.8043E+08	1.4923E+09

Table 3
The distortion using a real image: "Lena".

Method	k		
	64	128	256
FGKM	1.8015E+07	1.3766E+07	1.0632E+07
MGKM	1.8011E+07	1.3736E+07	1.0503E+07
MFGKM (T=1)	1.7836E+07	1.3755E+07	1.0567E+07
MFGKM (T=2)	1.7962E+07	1.3676E+07	1.0463E+07
MFGKM+MCS (T=1, s=2)	1.7992E+07	1.3823E+07	1.0601E+07
MFGKM+MCS (T=1, s=4)	1.7947E+07	1.3829E+07	1.0633E+07
MFGKM+MCS (T=2, s=2)	1.8099E+07	1.3799E+07	1.0549E+07
MFGKM+MCS (T=2, s=4)	1.7901E+07	1.3907E+07	1.0720E+07

Table 4
The computing time (in seconds) using three images: "Peppers," "Lena," and "Baboon".

Method	k		
	64	128	256
FGKM	5760.88	11645.73	23602.02
MGKM	11780.34	23484.63	47003.75
MFGKM (T=1)	4580.95	7915.17	14096.25
MFGKM (T=2)	8953.45	15344.17	26810.95
MFGKM+MCS (T=1, s=2)	2458.95	4207.47	7385.50
MFGKM+MCS (T=1, s=4)	1318.23	2174.05	3817.51
MFGKM+MCS (T=2, s=2)	4787.77	8045.67	13856.45
MFGKM+MCS (T=2, s=4)	2647.86	4358.56	7582.75

number of axes (n_a) used for the candidate construction algorithm. In this example, n_a=3 is used. Tables 1 and 2 give the execution time and numbers of distance calculations, respectively, for FGKM, MGKM, MFGKM with two values of T, and MFGKM+MCS with various values of T and s. Table 3 presents the distortions for these methods. From these tables, we may find

Table 5
The number of distance calculations using three images: "Peppers," "Lena," and "Baboon".

Method	k		
	64	128	256
FGKM	1.5635E+11	3.1944E+11	6.5395E+11
MGKM	3.0959E+11	6.2608E+11	1.2681E+12
MFGKM (T=1)	6.7765E+09	1.0903E+10	1.8025E+10
MFGKM (T=2)	1.2007E+10	1.7963E+10	2.7987E+10
MFGKM+MCS (T=1, s=2)	3.7491E+09	6.0428E+09	1.0521E+10
MFGKM+MCS (T=1, s=4)	2.3322E+09	3.6781E+09	6.5792E+09
MFGKM+MCS (T=2, s=2)	6.5968E+09	9.9650E+09	1.5574E+10
MFGKM+MCS (T=2, s=4)	4.2582E+09	6.1871E+09	9.8235E+09

Table 6
The distortion using three images: "Peppers," "Lena," and "Baboon".

Method	k		
	64	128	256
FGKM	1.3272E+08	1.1251E+08	9.5270E+07
MGKM	1.3272E+08	1.1239E+08	9.5111E+07
MFGKM (T=1)	1.3234E+08	1.1242E+08	9.5171E+07
MFGKM (T=2)	1.3229E+08	1.1220E+08	9.4934E+07
MFGKM+MCS (T=1, s=2)	1.3216E+08	1.1242E+08	9.5117E+07
MFGKM+MCS (T=1, s=4)	1.3212E+08	1.1249E+08	9.5099E+07
MFGKM+MCS (T=2, s=2)	1.3253E+08	1.1249E+08	9.4901E+07
MFGKM+MCS (T=2, s=4)	1.3351E+08	1.1260E+08	9.5142E+07

Table 7
The computing time (in seconds) using the Statlog data set.

Method	k		
	64	128	256
FGKM	151.02	309.92	639.16
MGKM	296.11	593.49	1196.19
MFGKM (T=1)	41.09	69.67	121.58
MFGKM (T=2)	78.22	129.52	215.38
MFGKM+MCS (T=1, s=2)	26.13	43.22	73.47
MFGKM+MCS (T=1, s=4)	13.75	23.41	39.67
MFGKM+MCS (T=2, s=2)	49.38	80.55	130.88
MFGKM+MCS (T=2, s=4)	28.11	44.50	71.84

Table 8
The number of distance calculations using the Statlog data set.

Method	k		
	64	128	256
FGKM	2.8225E+09	5.8476E+09	1.2141E+10
MGKM	5.4655E+09	1.1157E+10	2.2716E+10
MFGKM (T=1)	1.6699E+08	2.8371E+08	5.5569E+08
MFGKM (T=2)	2.5861E+08	4.0867E+08	7.2311E+08
MFGKM+MCS (T=1, s=2)	1.0964E+08	1.8160E+08	3.3239E+08
MFGKM+MCS (T=1, s=4)	6.1129E+07	1.1344E+08	2.0512E+08
MFGKM+MCS (T=2, s=2)	1.6623E+08	2.5874E+08	4.2855E+08
MFGKM+MCS (T=2, s=4)	1.1112E+08	1.7196E+08	2.7818E+08

that our method MFGKM with T=2 has the least distortion; while MFGKM+MCS with T=1 and s=4 gives the least computing time and number of distance calculations. The computing time of MFGKM with T=2 is almost two times of that for MFGKM with

T=1. This fact explains that the computational complexity of MFGKM is mainly from finding the starting cluster center at each step of cluster split. From Table 3, we may also find that compared to MFGKM+MCS with s=4, the computing time of MFGKM+MCS with s=2 is almost doubled. This is because that compared to MFGKM+MCS with s=2, the computational complexity of determining starting cluster centers for MFGKM+MCS with s=4 is reduced by a factor of 2. This phenomenon does confirm that the major computational complexity of MFGKM+MCS is also from the process of finding starting cluster centers at each step of cluster split. From Table 3, we will discover that the distortion of MFGKM+MCS with T=2 and s=2 or 4 is higher than that of MFGKM with T=2. This fact says that compared with MCS, the cluster center selection algorithm may have the better performance of selecting initial cluster centers at each step of cluster split.

Compared to MGKM, our method MFGKM with T=1 can reduce the computing time and number of distance calculations by a factor of 5.66–7.66 and 37.28–52.83, respectively, with the average reduction of distortion of 30760. Compared with FGKM, our method MFGKM+MCS with T=1 and s=2 can reduce the computing time and number of distance calculations by a factor of 4.62–6.49 and 34.56–46.78, respectively, with the average increase of distortion of 800. It is noted that compared to FGKM, MGKM can obtain less distortion with more computing time.

4.3. Example 3: a large data set generated from three real images: "Peppers," "Lena," and "Baboon."

In this example, the data set consists of 49,152 data points with d=16. For this large data set, the value of n_a is 6. Table 4 shows the computing time for FGKM, MGKM, MFGKM with some values of T and MFGKM+MCS with various T and s; while Table 5 gives the numbers of distance calculations. Table 6 presents the distortions for these methods. From these tables, we may also find that MFGKM with T=2 can obtain the least distortion in average and MFGKM+MCS with T=1 and s=4 gives the least computing time and number of distance calculations. From table 4, we may also find that the computing time of MFGKM with T=2 and MFGKM+MCS with T=2 and s=4 is almost doubled by MFGKM with T=1 and MFGKM+MCS with T=2 and s=2, respectively. This is due to the major computational complexities of MFGKM and MFGKM+MCS are from the processes of selecting the starting points at each cluster split.

Compared to MGKM, our method MFGKM with T=2 can reduce the computing time and number of distance calculations by a factor of 1.31–1.75 and 25.78–45.31, respectively, with the reduction of distortion from 176,455 to 433,385. Compared to MGKM, our method MFGKM+MCS with T=1 and s=2 can reduce the computing time and number of distance calculations by a factor of 4.79–6.36 and 82.58–120.53, respectively, with the average reduction of distortion of 175,189.

Table 9
The distortion using the Statlog data set.

Method	k		
	64	128	256
FGKM	5.8630E+06	4.6324E+06	3.6639E+06
MGKM	5.8297E+06	4.5840E+06	3.5746E+06
MFGKM (T=1)	5.8283E+06	4.6084E+06	3.6013E+06
MFGKM (T=2)	5.8181E+06	4.5738E+06	3.5799E+06
MFGKM+MCS (T=1, s=2)	5.8829E+06	4.5956E+06	3.5891E+06
MFGKM+MCS (T=1, s=4)	5.9136E+06	4.6349E+06	3.6108E+06
MFGKM+MCS (T=2, s=2)	5.9350E+06	4.6180E+06	3.6129E+06
MFGKM+MCS (T=2, s=4)	5.9389E+06	4.7080E+06	3.6810E+06

4.4. 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 [24]. The value of each coordinate for a data point is the range of 0–255. The more detailed description regarding this data set can be found in Ref. [24]. The value of n_a for the Statlog data set is 6. Table 7 shows the computing time for FGKM, MGKM, MFGKM, and MFGKM+MCS; while Table 8 gives the numbers of distance calculations. Table 9 presents the distortions for these methods. From these tables, we still find that MFGKM with T=2 generates the least distortion and MFGKM+MCS with T=1 and s=4 gives the least computing time and number of distance calculations. From Table 9, we may find that MFGKM with T=2 can reduce the distortion of MFGKM+MCS with T=2 and s=2 by 0.92–1.97%.

Compared to MGKM, our method MFGKM with T=2 can reduce the computing time and number of distance calculations by a factor of 3.78–5.55 and 21.13–31.41, respectively, with the average reduction of distortion of 5,487. Compared to FGKM, our method MFGKM+MCS with T=1 and s=2 can reduce the computing time by a factor of 5.78–8.70 with the average reduction of distortion of 30,564. Compared to FGKM, MGKM can reduce significantly the distortion with much more computing time for this high dimensional data set.

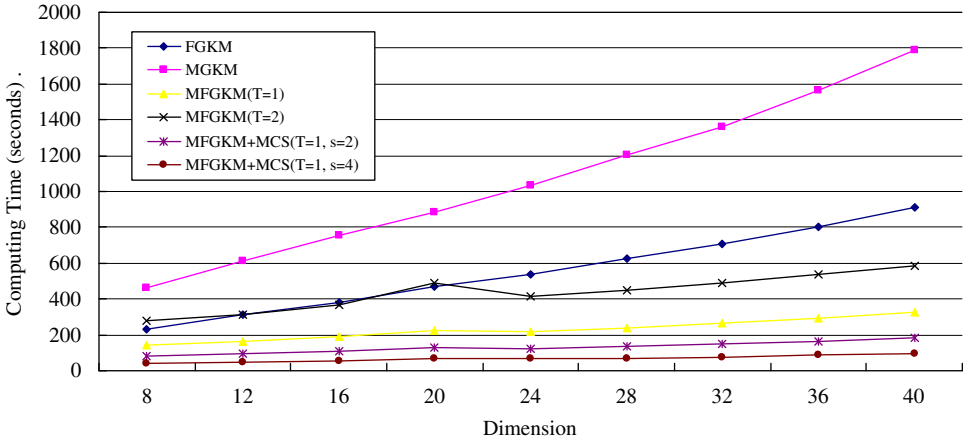

Fig. 2. The computing time for data sets from example 5 with N=10,000 and k=128.

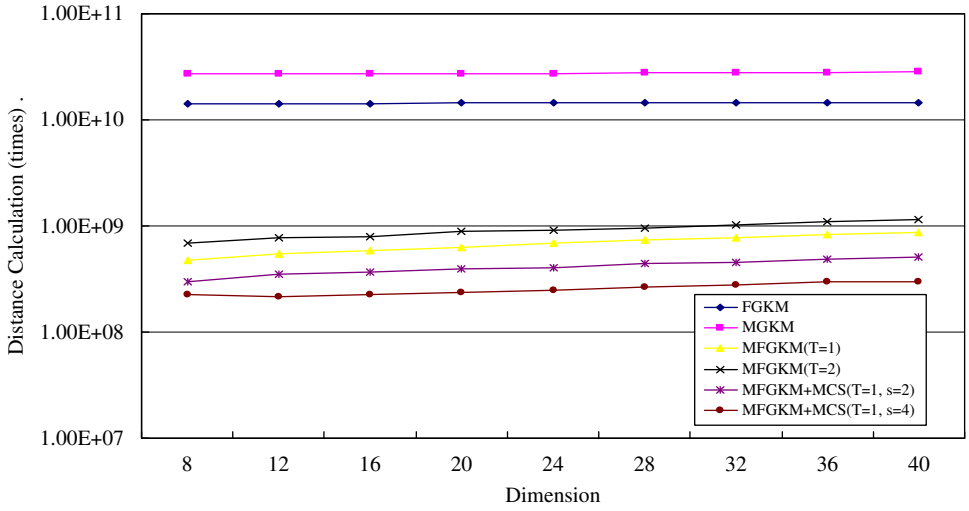

Fig. 3. The number of distance calculations for data sets from example 5 with N=10,000 and k=128.

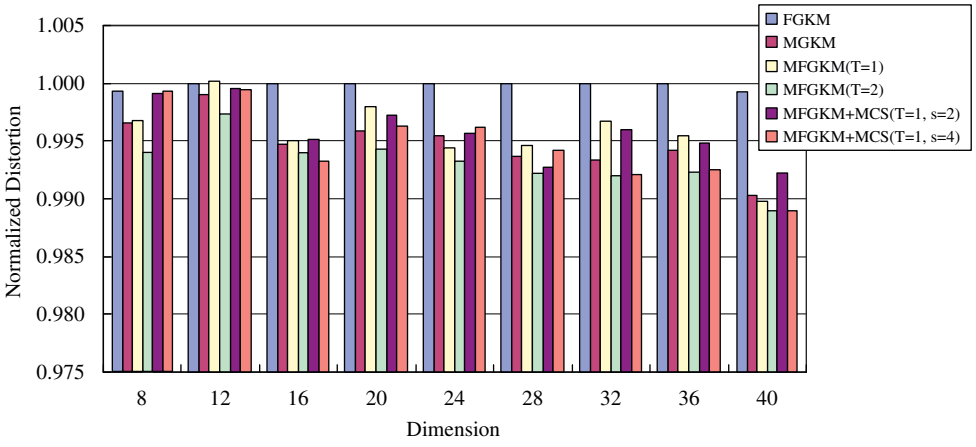

Fig. 4. The distortion for data sets from example 5 with N=10,000 and k=128.

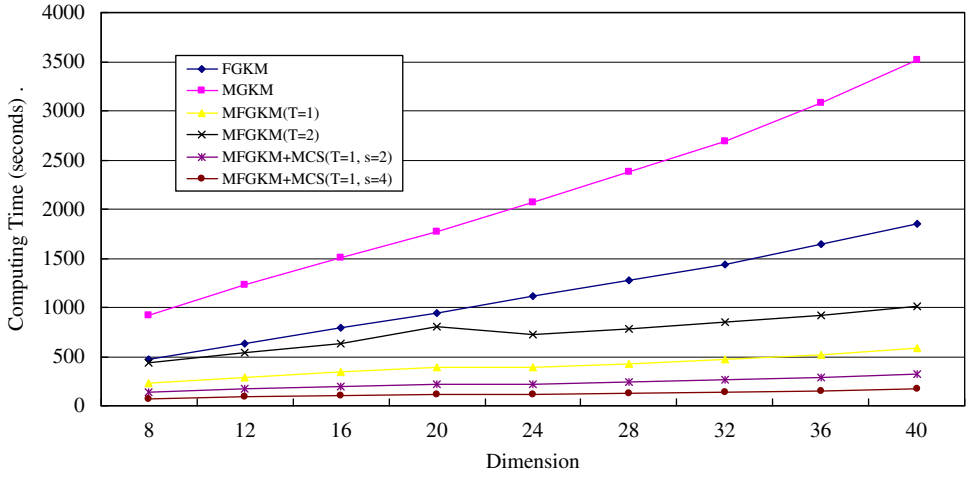

Fig. 5. The computing time for data sets from example 5 with N=10,000 and k=256.

4.5. *Example 5: data sets generated from the Gauss Markov source*

In this example, each data set is obtained from the Gauss Markov source. The values of n_a used for these data sets are 6. Figs. 2 and 5 present the computing time for k=128 and 256, respectively. Figs. 3 and 6 show the numbers of distance calculations with k=128 and 256; while Figs. 4 and 7 give the normalized distortions with k=128 and 256. From Figs. 4 and 7, we may find that MFGKM with T=2 has the least distortion. Compared to FGKM, our method MFGKM with T=2 can reduce

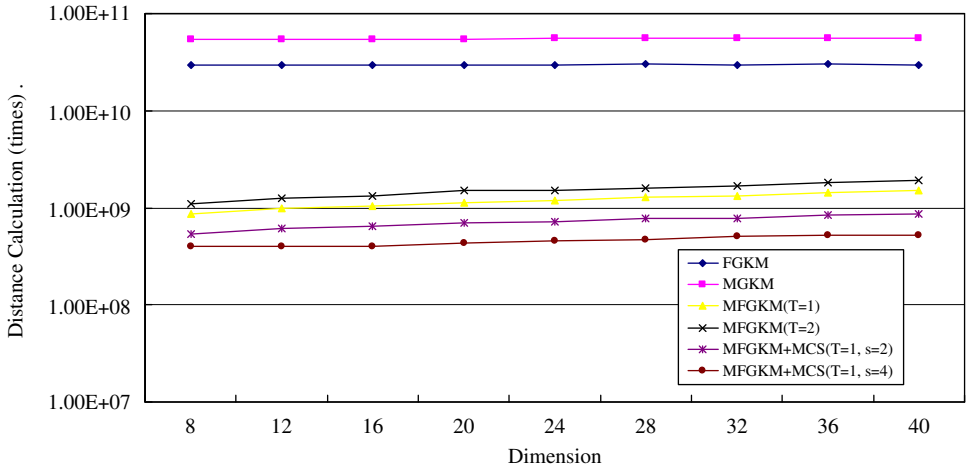

Fig. 6. The number of distance calculations for data sets from example 5 with N=10,000 and k=256.

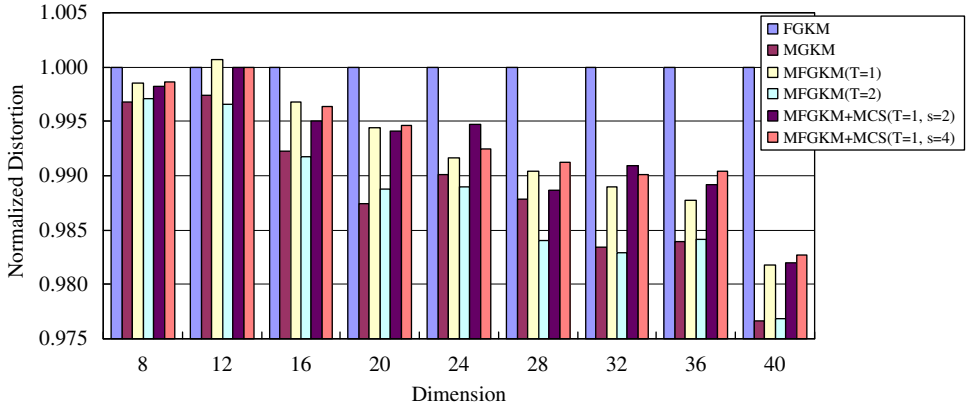

Fig. 7. The distortion for data sets from example 5 with N=10,000 and k=256.

distortion more significantly for a data set with higher dimension. Compared with MGKM, MFGKM with T=2 can reduce the computing and number of distance calculations more significantly, if a data set with higher dimension is used. Compared with FGKM, our method MFGKM+MCS can reduce the computing time, distortion, and number of distance calculations dramatically for the data set with higher dimension. That is, the performances of our proposed methods are more remarkable if data sets with higher dimension are used. From Figs. 4 and 7, we can find that MGKM and MFGKM with T=2 may obtain the same distortion for a high dimensional data set, although MFGKM with T=2 needs the much less computing time. Compared to FGKM, MFGKM+MCS can reduce the distortion for a high dimensional data set (Fig. 7).

5. Conclusions

In this paper, we develop MFGKM and MFGKM+MCS to speed up global k-means clustering through using a data point's cluster membership information. A set of inequalities is also developed to reduce the computational complexity of determining a starting point for a new cluster center of global k-means clustering. Our proposed method MFGKM with T=2 can obtain the least distortion and MFGKM+MCS with T=1 and s=4 provides the least computing time and number of distance calculation. Compared to MGKM, our method MFGKM with T=2 can reduce the computing time and number of distance calculations by a factor of 1.31–1.75 and 25.78–45.31, respectively, with the reduction of distortion from 176,455 to 433,385 for the data set generated from three images: "Peppers," "Lena," and "Baboon." Compared with MGKM, our method MFGKM+MCS with T=1 and s=2 can reduce the computing time and number of distance calculations by a factor 4.79–6.36 and 82.58–120.53, respectively, with the average distortion reduction of 175,189 for the same data set. The performances of our proposed methods are more remarkable when a data set with higher dimension is divided into more clusters.

References

[1] A. Gersho, R.M. Gray, Vector Quantization and Signal Compression, Kluwer Academic Publishers, Boston MA, 1991.
[2] Y.C. Liaw, J.Z.C. Lai, W. Lo, Image restoration of compressed image using classified vector quantization, Pattern Recognition 35 (2) (2002) 181–192.
[3] J. Foster, R.M. Gray, M.O. Dunham, Finite state vector quantization for waveform coding, IEEE Trans. Inf. Theory 31 (3) (1985) 348–359.
[4] J.Z.C. Lai, Y.C. Liaw, W. Lo, Artifact reduction of JPEG coded images using mean-removed classified vector quantization, Signal Process. 82 (10) (2002) 1375–1388.
[5] S. Theodoridis, K. Koutroumbas, Pattern Recognition, second ed., Academic Press, New York, 2003.
[6] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, Advances in Knowledge Discovery and Data Mining, MIT Press, Boston MA, 1996.
[7] D. Liu, F. Kubala, Online speaker clustering, Proc. IEEE Conf. Acoust., Speech, Signal Process. 1 (2004) 333–336.

[8] P. Hojen-Sorensen, N. de Freitas, T. Fog, On-line probabilistic classification with particle filters, Proc. IEEE Signal Process. Soc. Workshop 1 (2000) 386–395.
[9] M. Eirinaki, M. Vazirgiannis, Web mining for web personalization, ACM Trans. Internet Technol. 3 (2003) 1–27.
[10] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, A. Wu, An efficient *k*-means clustering algorithm: analysis and implementation, IEEE Trans. Pattern Anal. Mach. Intell. 24 (7) (2002) 881–892.
[11] J.Z.C. Lai, C.C. Lue, Fast search algorithms for VQ codebook generation, J. Visual Commun. Image Representation 7 (2) (1996) 163–168.
[12] A. Likas, M. Vlassis, J. Verbeek, The global *k*-means clustering algorithm, Pattern Recognition 35 (2) (2003) 451–461.
[13] A.M. Bagirov, Modified global *k*-means algorithm for minimum sum-of-squares clustering problems, Pattern Recognition 41 (10) (2008) 3192–3199.
[14] J.Z.C. Lai, Y.C. Liaw, J. Liu, A fast VQ codebook generation algorithm using codeword displacement, Pattern Recognition 41 (1) (2008) 315–319.
[15] D. Pelleg, A. Moore, Accelerating exact *k*-means clustering, in: Proceedings of ACM SIGKDD International Conference on Knowledge and Data Mining, August 1999, pp. 277–281.
[16] J.Z.C. Lai, Y.C. Liaw, Improvement of the *k*-means clustering filtering algorithm, Pattern Recognition 41 (12) (2008) 3677–3681.
[17] S.W. Ra, J.K. Kim, Fast mean-distance-ordered partial codebook search algorithm for image vector quantization, IEEE Trans. Circuits Syst. II 40 (9) (1993) 576–579.
[18] J.Z.C. Lai, Y.C. Liaw, Fast-searching algorithm for vector quantization using projection and triangular inequality, IEEE Trans. Image Process. 13 (12) (2004) 1554–1558.
[19] J.Z.C. Lai, Y.C. Liaw, J. Liu, Fast *k*-nearest-neighbor search based on projection and triangular inequality, Pattern Recognition 40 (2) (2007) 351–359.
[20] W.H. Cooley, P.R. Lohnes, Multivariate Data Analysis, Wiley, New York, 1971.
[21] S. Jana, P. Moulin, Optimality of KLT for high-rate transform coding of Gaussian vector-scale mixtures: applications to reconstruction, estimation, and classification, IEEE Trans. Inf. Theory 52 (9) (2006) 4049–4067.
[22] S.C. Tai, C.C. Lai, Y.C. Lin, Two fast nearest neighbor searching algorithms for image vector quantization, IEEE Trans. Commun. 40 (1996) 1623–1628.
[23] A. Sharma, K.K. Paliwal, Fast principal component analysis using fixed-point algorithm, Pattern Recognition Lett. 28 (10) (2007) 1151–1155.
[24] 〈http://www.ics.uci/∼mlearn/MLRespository.html〉.

About the Author—JIM Z.C. LAI 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 professor. 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.

About the Author—TSUNG-JEN HUANG received his B.S. and M.S. degrees in Information Engineering and Computer Science from Feng-Chia University, Taiwan, in 1992 and 1994, respectively. From 1994 to 2001, he was a Ph.D. student in the Department of Computer Science from National Chiao Tung University, Hsinchu, Taiwan. Since 2001, he has been an engineer with Industrial Technology Research Institute, Hsinchu, Taiwan. Moreover he is a Ph.D. student in Computer Science at National Taiwan Ocean University, Keelung, Taiwan. His current interests are in image processing, data mining and clustering algorithms.