An Entropy Weighting k-Means Algorithm for Subspace Clustering of High-Dimensional Sparse Data

Liping Jing, Michael K. Ng, and Joshua Zhexue Huang

Abstract—This paper presents a new k-means type algorithm for clustering high-dimensional objects in subspaces. In high-dimensional data, clusters of objects often exist in subspaces rather than in the entire space. For example, in text clustering, clusters of documents of different topics are categorized by different subsets of terms or keywords. The keywords for one cluster may not occur in the documents of other clusters. This is a data sparsity problem faced in clustering high-dimensional data. In the new algorithm, we extend the k-means clustering process to calculate a weight for each dimension in each cluster and use the weight values to identify the subsets of important dimensions that categorize different clusters. This is achieved by including the weight entropy in the objective function that is minimized in the k-means clustering process. An additional step is added to the k-means clustering process to automatically compute the weights of all dimensions in each cluster. The experiments on both synthetic and real data have shown that the new algorithm can generate better clustering results than other subspace clustering algorithms. The new algorithm is also scalable to large data sets.

Index Terms—k-means clustering, variable weighting, subspace clustering, text clustering, high-dimensional data.

1 Introduction

High-dimensional data is a phenomenon in real-world data mining applications. Text data is a typical example. In text mining, a text document is viewed as a set of pairs $< t_i, f_i >$, where t_i is a term or word, and f_i is a measure of t_i, for example, the frequency of t_i in the document. The total number of unique terms in a text data set represents the number of dimensions, which is usually in the thousands. High-dimensional data occurs in business as well. In retail companies, for example, for effective supplier relationship management (SRM), suppliers are often categorized in groups according to their business behaviors. The supplier's behavior data is high dimensional because thousands of attributes are used to describe the supplier's behaviors, including product items, ordered amounts, order frequencies, product quality, and so forth.

Sparsity is an accompanying phenomenon of high-dimensional data. In text data, documents related to a particular topic, for instance, sport, are categorized by one subset of terms. The terms describing sport may not occur in the documents describing music. This implies that f_i is zero for the subset of terms in the documents describing music, and vice versa. Such situation also occurs in supplier categorization. A group of suppliers are categorized by the subset of product items supplied by the suppliers. Other suppliers who did not supply these product items have zero order amount for them in the behavior data [1].

Clearly, clustering of high-dimensional sparse data requires special treatment [2], [3], [4], [5]. This type of clustering methods is referred to as subspace clustering, aiming at finding clusters from subspaces of data instead of the entire data space. In a subspace clustering, each cluster is a set of objects identified by a subset of dimensions and different clusters are represented in different subsets of dimensions. The major challenge of subspace clustering, which makes it distinctive from traditional clustering, is the simultaneous determination of both cluster memberships of objects and the subspace of each cluster.

Cluster memberships are determined by the similarities of objects measured with respect to subspaces. According to the ways that the subspaces of clusters are determined, subspace clustering methods can be divided into two types. The first type is to find out the exact subspaces of different clusters (see, for instance, [6], [7], [8], [9], [10], [11], [12]). We call these methods as *hard subspace clustering*. The second type is to cluster data objects in the entire data space but assign different weighting values to different dimensions of clusters in the clustering process, based on the importance of the dimensions in identifying the corresponding clusters (see, for instance, [13], [14], [15], [16], [17], [18], [19], [20], [21], [22]). We call these methods *soft subspace clustering*.

In this paper, we present a new k-means type algorithm for soft subspace clustering of large high-dimensional sparse data. We consider that different dimensions make different contributions to the identification of objects in a cluster. The difference of contribution of a dimension is represented as a weight that can be treated as the degree of the dimension in contribution to the cluster. In subspace clustering, the decrease of the weight entropy in a cluster implies the increase of certainty of a subset of dimensions

- *L. Jing is with the Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong. E-mail: lpjing@eti.hku.hk.*
- *M.K. Ng is with the Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong. E-mail: mng@math.hkbu.edu.hk.*
- *J.Z. Huang is with E-Business Technology Institute, The University of Hong Kong, Pokfulam Road, Hong Kong. E-mail: jhuang@eti.hku.hk.*

Manuscript received 16 May 2006; revised 10 Jan. 2007; accepted 9 Apr. 2007; published online 18 Apr. 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0255-0506.
Digital Object Identifier no. 10.1109/TKDE.2007.1048.

with larger weights in determination of the cluster. Therefore, in the clustering process, we simultaneously minimize the within cluster dispersion and maximize the negative weight entropy to stimulate more dimensions to contribute to the identification of a cluster. This can avoid the problem of identifying clusters by a few dimensions with sparse data. A new formula for computing a dimension weight is derived and added to the k-means clustering process as an additional step in each iteration, so the cluster memberships of objects and the weights of dimensions in each cluster can be obtained simultaneously. The results of extensive experiments on both synthetic and real data have demonstrated that the new algorithm outperformed the other subspace clustering algorithms in clustering accuracy. It was also effective in clustering sparse data and scalable in clustering large data with respect to the number of dimensions and the number of clusters.

The rest of the paper is organized as follows: In Section 2, we review some related work. In Section 3, we present the new k-means type subspace clustering algorithm. In Section 4, we show simulation results of the new algorithm on synthetics data. In Section 5, we present experiment results on two real data sets. We give concluding remarks on the new algorithm in Section 6.

2 Related Work

Subspace clustering seeks to group objects into clusters on subsets of dimensions or attributes of a data set. It pursues two tasks, identification of the subsets of dimensions where clusters can be found and discovery of the clusters from different subsets of dimensions. According to the ways with which the subsets of dimensions are identified, we can divide subspace clustering methods into two categories. The methods in the first category determine the exact subsets of dimensions where clusters are discovered. We call these methods *hard subspace clustering*. The methods in the second category determine the subsets of dimensions according to the contributions of the dimensions in discovering the corresponding clusters. The contribution of a dimension is measured by a weight that is assigned to the dimension in the clustering process. We call these methods *soft subspace clustering* because every dimension contributes to the discovery of clusters, but the dimensions with larger weights form the subsets of dimensions of the clusters. The method in this paper falls in the second category.

2.1 Hard Subspace Clustering

The subspace clustering methods in this category can be further divided into bottom-up and top-down subspace search methods [23]. The bottom-up methods for subspace clustering consist of the following main steps:

1. dividing each dimension into intervals and identifying the dense intervals in each dimension,
2. from the interactions of the dense intervals, identifying the dense cells in all two dimensions,
3. from the intersections of 2D dense cells and the dense intervals of other dimensions, identifying the dense cells in all three dimensions and repeating this process until all dense cells in all k dimensions are identified, and
4. merging the adjacent dense cells in the same subsets of dimensions to identify clusters.

Examples of the bottom-up methods include *CLIQUE* [6], *ENCLUS* [24], and *MAFIA* [25].

PROCLUS [7] is a representative subspace clustering algorithm based on the top-down methods. It starts with a set of initial cluster centers discovered from a small data sample based on the k-medoid clustering method [26]. The initial centers are made as apart as possible to each other. For each center, a set of data points within a distance δ to the center are identified as the center *locality* L_i. Here, δ is the minimal distance between the center and other centers. For each L_i, the average distance between the points in L_i and the center is computed in each dimension. The subset of dimensions whose average distances are smaller than the average distance of all dimensions is considered as the candidate subspace for cluster i. After all candidate subspaces are identified, the clusters are discovered from the subspaces using the distance measures on subsets of dimensions. Using *PROCLUS*, the user needs to specify the number of clusters k and the average number of cluster dimensions l. Other top-down methods include *ORCLUS* [8], FINDIT [27], and δ-Clusters [28].

Local Dimensionality Reduction (*LDR*) [9], [29], like *PROCLUS*, projects each cluster on its associated subspace, which is generally different from the subspace associated with another cluster. The efficacy of this method depends on how the clustering problem is addressed in the first place in the original feature space. A potentially serious problem with such a technique is the lack of data to locally perform PCA on each cluster to derive the principal components, therefore, it is inflexible in determining the dimensionality of data representation. Moreover, for clustering purposes, the new dimensions may be difficult to interpret, making it hard to understand clusters in relation to the original space. Additionally, *LDR* has high computational complexity, $O(nm^2k)$ [9], where n is the number of objects, m is the dimensionality of the original feature space, and k is the number of clusters. Hence, *LDR* is restricted in applicability to specific problems such as high-dimensional text clustering.

A hierarchical subspace clustering approach with automatic relevant dimension selection, called *HARP*, was recently presented by Yip et al. [11]. *HARP* is based on the assumption that two objects are likely to belong to the same cluster if they are very similar to each other along many dimensions. Clusters are allowed to merge only if they are similar enough in a number of dimensions, where the minimum similarity and the minimum number of similar dimensions are controlled by two internal threshold parameters. Due to the hierarchical nature, the algorithm is intrinsically slow. Also, if the number of relevant dimensions per cluster is extremely low, the accuracy of *HARP* may drop as the basic assumption will become less valid due to the presence of a large amount of noise values in the data set.

2.2 Soft Subspace Clustering

Instead of identifying exact subspaces for clusters, this approach assigns a weight to each dimension in the clustering process to measure the contribution of the dimension in forming a particular cluster. In a clustering, every dimension contributes to every cluster, but contributions are different. The subspaces of the clusters can be identified by the weight values after clustering.

Variable weighting for clustering is an important research topic in statistics and data mining [13], [30], [31], [32], [15], [21]. However, the purpose is to select important variables for clustering.

Extensions to some variable weighting methods, for example, the k-means type variable weighting methods, can perform the task of subspace clustering. A number of algorithms in this direction have been reported recently [33], [19], [17], [18], [16], [20], [22]. Unlike variable selection in which a weight is assigned to a dimension for the entire data set, we assign a weight to each dimension for each cluster. As such, different clusters have different sets of weight values. To retain the scalability, the k-means clustering process is adopted in these new subspace clustering algorithms. In each iteration, an additional step is added to compute the weight values. The differences of these algorithms lie in the weight formulas that result from different objective functions to be minimized in the clustering process.

The direct extension to the k-means type variable, weighting algorithm [21] for variable selection results from the minimization of the following objective function [17], [16]:

$$J_1(Z,W,\Lambda)=\sum_{l=1}^{k}\sum_{j=1}^{n}w_{lj}^{\eta}\sum_{i=1}^{m}\lambda_{li}^{\beta}(z_{li}-x_{ji})^2 \tag{1}$$

subject to

$$\begin{cases}\sum\limits_{l=1}^{k}w_{lj}=1, & 1\le j\le n,\\ 0\le w_{lj}\le 1, & 1\le l\le k, \quad 1\le j\le n,\\ \sum\limits_{i=1}^{m}\lambda_{li}=1, & 1\le l\le k,\\ 0\le \lambda_{li}\le 1, & 1\le l\le k, \quad 1\le i\le m.\end{cases}$$

Here, n, k, and m are the numbers of objects, clusters, and dimensions, respectively. $\beta\,(>1)$ and $\eta\,(\ge 1)$ are two parameters greater than 1. w_{lj} is the degree of membership of the jth object belonging to the lth cluster. λ_{li} is the weight for the ith dimension in the lth cluster. x_{ji} is value of the ith dimension of the jth object, and z_{li} is the value of the ith component of the lth cluster center. $\eta=1$ produces a hard clustering, whereas $\eta>1$ results in a fuzzy clustering.

There are three unknowns W, Z, and Λ that need to be solved. The first two can be solved in the same way as used in the standard k-means algorithm. The weight λ_{li} for each dimension in each cluster is solved with the following formula (it can be derived using the Lagrange multiplier technique):

$$\lambda_{li}=\frac{1}{\sum\limits_{t=1}^{m}\left[\frac{\sum_{j=1}^{n}w_{lj}^{\eta}(z_{li}-x_{ji})^2}{\sum_{j=1}^{n}w_{lj}^{\eta}(z_{lt}-x_{jt})^2}\right]^{1/(\beta-1)}}, \tag{2}$$

where w_{lj} and z_{li} represent the values in the current iteration.

We can observe that the weight value for a dimension in a cluster is inversely proportional to the dispersion of the values from the center in the dimension of the cluster. Since the dispersions are different in different dimensions of different clusters, the weight values for different clusters are different. The high weight indicates a small dispersion in a dimension of the cluster. Therefore, that dimension is more important in forming the cluster.

This subspace clustering algorithm has a problem in handling sparse data. If the dispersion of a dimension in a cluster happens to be zero, then the weight for that dimension is not computable. This situation occurs frequently in high-dimensional sparse data. To make the weights computable, a simple method is to add a small constant in the distance function to make all dispersions greater than zero [22], [34].

Domeniconi in her dissertation [33] and Domeniconi et al. [19] proposed the Locally Adaptive Clustering (LAC) algorithm for a minimization problem of the following objective function:

$$J_2(Z,\Lambda)=\sum_{l=1}^{k}\sum_{i=1}^{m}\lambda_{li}\ \exp(X_{li}), \tag{3}$$

where

$$X_{li}=\left[\frac{1}{n_l}\max_{i'}\left\{\sum_{z(j)=l}(z_{li'}-x_{ji'})^2\right\}-\frac{1}{n_l}\sum_{z(j)=l}(z_{li}-x_{ji})^2\right]$$

subject to

$$\begin{cases}\sum\limits_{i=1}^{m}\lambda_{li}^2=1, & 1\le l\le k,\\ 0\le\lambda_{li}\le 1, & 1\le l\le k, \quad 1\le i\le m.\end{cases}$$

Here, $z(j)=l$ means that the jth object is assigned to the lth cluster; otherwise, it is not assigned, and n_l is the number of objects in the lth cluster. There are two terms in X_{li}. The first term is the largest among average distances along each dimension in the lth cluster, and the second term is the average distance between the cluster center z_l and the objects in the lth cluster along the ith dimension. In the algorithm, each dimension weight λ_{li} can be computed as follows:

$$\lambda_{li}=\frac{\exp(\tau\times X_{li})}{\sqrt{\sum\limits_{t=1}^{m}\exp(\tau\times 2\times X_{lt})}}, \tag{4}$$

where τ is a parameter that is used to maximize/minimize the influence of X_{li} on λ_{li}. When $\tau=0$, all the dimension weights will have the same value $1/m$, that is, ignoring any difference between X_{li}. On the other hand, when τ is large, a change of X_{li} will be exponentially reflected in λ_{li}. However, the objective function in (3) is not differentiable because of a maximum function. The convergence of the algorithm is proved by replacing the largest average distance in each dimension with a fixed constant value.

Friedman and Meulman recently published the Clustering Objects on Subsets of Attributes (COSA) algorithm for subspace clustering, which has generated a lot of interest [20]. The method is to assign a weight to every dimension in each cluster and determine the clusters by optimizing the following objective function:

$$J_3(Z,\{\Lambda_l\}_{l=1}^k) = \sum_{l=1}^{k}\frac{\Lambda_l}{n_l^2}\sum_{z(j)=z(j')=l}\left[\sum_{i=1}^{m}(\lambda_{li}d_{jj'i}+\alpha\lambda_{li}\log\lambda_{li})+\alpha\log m\right]. \quad (5)$$

Here, α is a positive parameter to control the strength of the incentive for subspace clustering on more dimensions, n_l is the number of objects assigned to the lth cluster, Λ_l is the weight vector for the lth cluster for regulating the cluster size, λ_{li} is the weight of the ith feature in the lth cluster, $d_{jj'i}$ is the distance between the jth and the j'th objects along the ith dimension. For instance, the distance is given by

$$d_{jj'i} = \frac{|x_{ji}-x_{j'i}|}{\frac{1}{n^2}\sum_{j_1=1}^{n}\sum_{j_2=1}^{n}|x_{j_1i}-x_{j_2i}|}.$$

In order to minimize (5) and find the solution clusters efficiently, Friedman and Meulman proposed to use an iterative approach to build a weighted dissimilarity matrix among objects. Then, a hierarchical clustering method-based nearest neighbors is used to cluster this matrix. The computational process of *COSA* may not be scalable to large data sets. Its computational complexity of building the weighted dissimilarity matrix is $O(hnmL+n^2m)$ (n is the number of objects, m is the number of dimensionality, L is a predefined parameter to find L *nearest neighbors* objects of a given object, and h is the number of iterations), where the first term of the complexity is for calculating weights of all dimensions for each object, and the second term is for creating the matrix. In other words, *COSA* may not be practical for large-volume and high-dimensional data.

3 Entropy Weighting *k*-Means

In this section, we present a new k-means type algorithm for soft subspace clustering of high-dimensional sparse data. In the new algorithm, we consider that the weight of a dimension in a cluster represents the probability of contribution of that dimension in forming the cluster. The entropy of the dimension weights represents the certainty of dimensions in the identification of a cluster. Therefore, we modify the objective function (1) by adding the weight entropy term to it so that we can simultaneously minimize the within cluster dispersion and maximize the negative weight entropy to stimulate more dimensions to contribute to the identification of clusters. In this way, we can avoid the problem of identifying clusters by few dimensions in sparse data.

The new objective function is written as follows:

$$F(W,Z,\Lambda) = \sum_{l=1}^{k}\left[\sum_{j=1}^{n}\sum_{i=1}^{m}w_{lj}\lambda_{li}(z_{li}-x_{ji})^2+\gamma\sum_{i=1}^{m}\lambda_{li}\log\lambda_{li}\right] \quad (6)$$

subject to

$$\begin{cases}\sum_{l=1}^{k}w_{lj}=1, & 1\le j\le n, \quad 1\le l\le k, \quad w_{lj}\in\{0,1\}\\ \sum_{i=1}^{m}\lambda_{li}=1, & 1\le l\le k, \quad 1\le i\le m, \quad 0\le\lambda_{li}\le 1.\end{cases}$$

The first term in (6) is the sum of the within cluster dispersions, and the second term the negative weight entropy. The positive parameter γ controls the strength of the incentive for clustering on more dimensions.

Next, we present the entropy weighting k-means algorithm (*EWKM*) to solve the above minimization problem.

3.1 *EWKM* Algorithm

Minimization of F in (6) with the constraints forms a class of constrained nonlinear optimization problems whose solutions are unknown. The usual method toward optimization of F is to use the partial optimization for Λ, Z, and W. In this method, we first fix Z and Λ and minimize the reduced F with respect to W. Then, we fix W and Λ and minimize the reduced F with respect to Z. Afterward, we fix W and Z and minimize the reduced F to solve Λ.

We can extend the standard k-means clustering process to minimize F by adding an additional step in each iteration to compute weights Λ for each cluster. The formula for computing Λ is given in the following theorem:

Theorem 1. *Given matrices W and Z are fixed, F is minimized if*

$$\lambda_{lt} = \frac{\exp\left(\frac{-D_{lt}}{\gamma}\right)}{\sum_{i=1}^{M}\exp\left(\frac{-D_{li}}{\gamma}\right)}, \quad (7)$$

where

$$D_{lt} = \sum_{j=1}^{n}w_{lj}(z_{lt}-x_{jt})^2.$$

Proof. We use the Lagrangian multiplier technique to obtain the following unconstrained minimization problem:

$$\min F_1(\{\lambda_{li}\},\{\delta_l\}) = \sum_{l=1}^{k}\left[\sum_{j=1}^{n}\sum_{i=1}^{m}w_{lj}\lambda_{li}(z_{li}-x_{ji})^2+\gamma\sum_{i=1}^{m}\lambda_{li}\log\lambda_{li}\right]-\sum_{l=1}^{k}\delta_l\left(\sum_{i=1}^{m}\lambda_{li}-1\right), \quad (8)$$

where $[\delta_1,\ldots,\delta_k]$ is a vector containing the Lagrange multipliers corresponding to the constraints. The optimization problem in (8) can be decomposed into k independent minimization problems:

$$\min F_{1l}(\lambda_{li},\delta_l) = \sum_{j=1}^{n}\sum_{i=1}^{m}w_{lj}\lambda_{li}(z_{li}-x_{ji})^2+\gamma\sum_{i=1}^{m}\lambda_{li}\log\lambda_{li}-\delta_l\left(\sum_{i=1}^{m}\lambda_{li}-1\right)$$

for $l=1,\ldots,k$. By setting the gradient of F_{1l} with respect to λ_{li} and δ_l to zero, we obtain

$$\frac{\partial F_{1l}}{\partial\delta_l} = \left(\sum_{i=1}^{m}\lambda_{li}-1\right)=0 \quad (9)$$

and

$$\frac{\partial F_{1l}}{\partial\lambda_{lt}} = \sum_{j=1}^{n}w_{lj}(z_{lt}-x_{jt})^2+\gamma(1+\log\lambda_{lt})-\delta_l=0. \quad (10)$$

From (10), we obtain

$$\lambda_{lt} = \exp\left(\frac{-D_{lt}-\gamma+\delta_l}{\gamma}\right) = \exp\left(\frac{\delta_l-\gamma}{\gamma}\right)\exp\left(\frac{-D_{lt}}{\gamma}\right), \tag{11}$$

where

$$D_{lt} = \sum_{j=1}^{n} w_{lj}(z_{lt}-x_{jt})^2$$

is interpreted as a measure of the dispersion of the data values on the tth dimension on the objects in the lth cluster. Substituting (11) into (9), we have

$$\sum_{i=1}^{m}\lambda_{li} = \sum_{i=1}^{m}\exp\left(\frac{\delta_l-\gamma}{\gamma}\right)\exp\left(\frac{-D_{li}}{\gamma}\right) = \exp\left(\frac{\delta_l-\gamma}{\gamma}\right)\sum_{i=1}^{m}\exp\left(\frac{-D_{li}}{\gamma}\right) = 1.$$

It follows that

$$\exp\left(\frac{\delta_l-\gamma}{\gamma}\right) = \frac{1}{\sum_{i=1}^{m}\exp\left(\frac{-D_{li}}{\gamma}\right)}.$$

Substituting this expression back to (11), we obtain

$$\lambda_{lt} = \frac{\exp\left(\frac{-D_{lt}}{\gamma}\right)}{\sum_{i=1}^{m}\exp\left(\frac{-D_{li}}{\gamma}\right)}. \tag{12}$$

□

Similarly to the k-means algorithm, given Z and Λ are fixed, W is updated as

$$\begin{cases} w_{lj}=1, & \text{if } \sum_{i=1}^{m}\lambda_{li}(z_{li}-x_{ji})^2 \le \sum_{i=1}^{m}\lambda_{ri}(z_{ri}-x_{ji})^2 \\ & \text{for } 1\le r\le k, \\ w_{lj}=0, & \text{otherwise.} \end{cases} \tag{13}$$

$w_{lj}=1$ means that the jth object is assigned to the lth cluster. If the distances between an object and two cluster centers are equal, the object is arbitrarily assigned to the cluster with the smaller cluster index number.

Given W and Λ are fixed, Z is updated as

$$z_{li} = \frac{\sum_{j=1}^{n} w_{lj}x_{ji}}{\sum_{j=1}^{n} w_{lj}} \quad \text{for } 1\le l\le k \text{ and } 1\le i\le m. \tag{14}$$

We note that (14) is independent of the parameter γ and the dimension weights λ_{li}.

The *EWKM* algorithm that minimizes (6), using (7), (13), and (14), is summarized as follows:

Algorithm—EWKM

Input: The number of clusters k and parameter γ;
Randomly choose k cluster centers and set all initial weights to $1/m$;
REPEAT
Update the partition matrix W by (13);
Update the cluster centers Z by (14);
Update the dimension weights Λ by (7);
UNTIL (the objective function obtains its local minimum value);

The input parameter γ is used to control the size of the weights as follows:

- $\gamma > 0$. In this case, according to (7), λ_{li} is inversely proportional to D_{li}. The smaller D_{li}, the larger λ_{li}, the more important the corresponding dimension.
- $\gamma = 0$. $\lambda_{li'}$ is equal to one, indicating that the index i' has the smallest value of $D_{li'}$. The other weights λ_{li} for $i \neq i'$ are equal to zero. Each cluster contains only one important dimension. It may not be desirable for high-dimensional data sets.
- $\gamma < 0$. In this case, according to (7), λ_{li} is proportional to D_{li}. The larger D_{li}, the larger λ_{li}. This is contradictory to the original idea of dimension weighting. Therefore, γ cannot be smaller than zero.

The simulation results of γ in subspace clustering will be shown in Section 4.

3.2 Convergency and Complexity Analysis

The *EWKM* algorithm converges in a finite number of iterations. To divide a data set into k clusters, the number of possible partitions is finite. We can show that each possible partition W only occurs once in the clustering process. Assume that $W^{h_1} = W^{h_2}$, where h is the iteration index and $h_1 \neq h_2$. We note that given W^h, we can compute the minimizer Z^h, which is independent of Λ^h according to (14). For W^{h_1} and W^{h_2}, we have the minimizers Z^{h_1} and Z^{h_2}, respectively. It is clear that $Z^{h_1} = Z^{h_2}$ since $W^{h_1} = W^{h_2}$. Using W^{h_1} and Z^{h_1}, and W^{h_2} and Z^{h_2}, we can compute the minimizers Λ^{h_1} and Λ^{h_2}, respectively, according to (7). It is clear that $\Lambda^{h_1} = \Lambda^{h_2}$. Therefore, we obtain

$$F(W^{h_1}, Z^{h_1}, \Lambda^{h_1}) = F(W^{h_2}, Z^{h_2}, \Lambda^{h_2}).$$

However, the sequence $F(\cdot,\cdot,\cdot)$ generated by the algorithm is strictly decreasing. Therefore, the *EWKM* algorithm converges in a finite number of iterations.

The *EWKM* algorithm is scalable to the number of dimensions, the number of objects, and the number of clusters. This is because *EWKM* only adds a new step to the k-means clustering process to calculate the dimension weights of each cluster. The runtime complexity can be analyzed as follows. We only consider the three major computational steps:

- **Partitioning the objects**. After initialization of the dimension weights of each cluster and the cluster centers, a cluster membership is assigned to each object. This process simply compares the summation of

  $$\sum_{i=1}^{m}\lambda_{li}(z_{li}-x_{ji})^2$$

  in (13) for each object in all k clusters. Thus, the complexity for this step is $O(mnk)$ operations.

- **Updating cluster centers.** Given the partition matrix W, updating cluster centers is to find the means of the objects in the same cluster. Thus, for k clusters, the computational complexity for this step is $O(mnk)$.
- **Calculating dimensions weights.** The last phase of this algorithm is to calculate the dimensions weights for all clusters based on the partition matrices W and Z. In this step, we only go through the whole data set once to update the dimensions weights. The computational complexity of this step is also $O(mnk)$.

If the clustering process needs h iterations to converge, the total computational complexity of this algorithm is $O(hmnk)$. This shows that the computational complexity increases linearly as the number of dimensions, objects, or clusters increases.

The proposed method is different from our previous work in [22], [34]. In this paper, we make use of the entropy term to assign the weights of different dimensions. It is not necessary to add a positive constant in the distance function to make all dispersions greater than zero. However, in [22], [34], a positive constant is required to be input in the distance function.

Finally, we compare the proposed algorithm with the COSA clustering algorithm. We remark that the weighting entropy term is also used in the objective functions in COSA [20]. The main difference between the proposed method and the COSA method is the minimizing algorithm for finding the cluster solutions. In the proposed method, we employ alternating minimization technique to update the weights, the cluster centers, and the membership weights. At each update, we compute the corresponding minimizer exactly. However, the COSA method tries to determine a good set of weights for calculating interpoint distances by approximately minimizing the criterion in (5) using the nearest neighbors only for each object. Then, weighted distances among objects are calculated and input to the hierarchical clustering algorithm.

4 Synthetic Data Simulations

The motivation for development of the *EWKM* algorithm is to cluster high-dimensional sparse data. To better understand the properties of the algorithm, synthetic data with controlled cluster structures and data sparsity were first used to investigate the relationships of the dispersion and weights of dimensions in each cluster, the behavior of parameter γ, and the performance of the algorithm on clustering accuracy in comparison with other clustering algorithms.

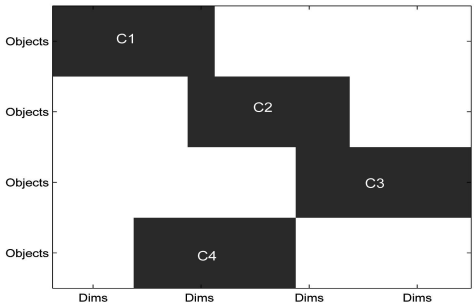

Fig. 1. The structure of a synthetic data set where the gray areas represent four clusters that are formed in different subspaces, and the white areas represent the dimensions where data entries are either zeros or random values.

4.1 Sparse Data Generation

The structure of a synthetic data set has the following characteristics: 1) it contains more than one cluster, 2) the data values of a cluster are concentrated on a subset of relevant dimensions, whereas other irrelevant dimensions contain mostly zero values with some random positive values, and 3) the relevant dimensions for different clusters can overlap. Fig. 1 illustrates an example of a synthetic data set with four clusters.

A similar process as given by Zait and Messatfa [35] was used to generate the synthetic data sets with different cluster structures. The parameters for controlling cluster structures are listed in Table 1.

The subspace ratio s is defined as

$$s = \frac{\sum_{l=1}^{k} m_l}{km}, \tag{15}$$

where m_l is the number of relevant dimensions in the lth cluster, and m is the total number of dimensions in the data set. The subspace ratio s determines the average size of the subspace of each cluster. The overlap ratio ρ determines the percentage of overlap dimensions between two clusters. The parameter ϵ controls the percentage of the positive values randomly generated for the irrelevant dimensions of a cluster.

TABLE 1
The Parameters for Generating a Synthetic Data Set

Parameter	Definition
k	Number of clusters
n	Number of objects
m	Number of dimensions
s	Subspace ratio
ρ	Overlap ratio
ϵ	Sparsity control for irrelevant dimensions
MINMU/MAXMU	Minimum/maximum value of the mean in a relevant dimension
MINV/MAXV	Minimum/maximum value of an entry in an irrelevant dimension

TABLE 2
The Algorithm for Generating Synthetic Data

Algorithm—*Generating a synthetic data set*

1. Initialize the random number generator and arrays as needed
   including standard variance σ and mean μ of the relevant dimensions
   in each cluster, and E, the indices array of relevant dimensions for each cluster;
2. Specify the numbers of clusters k, dimensions m and objects n;
   the parameters s, ρ and ϵ;
3. Determine the relevant dimensions for each cluster;
   //Generate the number of dimension for each cluster
   For $l = 1$ to k
   $m_l = random()$;
   where m_l in $[2, m]$;
   and $\sum_{l=1}^{k} m_l = s \times k \times m$;
   //Choose the dimensions for each cluster based on m_l
   For $l = 1$ to k
   If $l == 1$
   randomly chose m_1 dimensions for the first cluster;
   Else
   randomly chose $\rho \times m_l$ dimensions from the relevant dimensions of C_{l-1};
   randomly chose $(1 - \rho) \times m_l$ dimensions from the other dimensions;
   get a two-dimension array E;
4. Generate the means and variances for the relevant dimensions ;
   For $l = 1$ to k
   For $i = 1$ to m_l;
   $j = E(l, i)$;
   set the $\sigma_{l,j}$ for TYPE I, II, and III;
   randomly set mean $\mu_{l,j}$ in $[MINMU, MAXMU]$;
   //Guarrantee two clusters well separated in the respective subspace
   If the dimension j belongs to both C_l and C_{l-1};
   set the mean $\mu_{l,j}$ as $\mu_{l,j} \leq \mu_{l-1,j} + 2\sigma_{l,j}$;
5. Generate the data points for each cluster;
   For $l = 1$ to k
   //Specify the number of points for each cluster;
   $n_l = N/k$;
   //Generate the coordinates of the data points in the relevant dimensions;
   set the data points with normal distribution based on σ and μ;
   //Generate the coordinates of the data points in the irrelevant dimensions;
   specify $\epsilon \times (m - m_l) \times n_l$ as noise data;
   randomly set the coordinates of the noise data in $[MINV, MAXV]$;
   the other data are assumed as missing, set the values as zero;

In generating a cluster, the values of relevant dimensions conform to a normal distribution with given means and variances. The range of mean values is specified by parameters MINMU and MAXMU. For the random values of irrelevant dimensions, the value range is specified by parameters MINV and MAXV. The number of relevant dimensions m_l of a cluster is jointly determined by the parameters *subspace ratio* s and *overlap ratio* ρ, and a random number between 2 and m. The number of irrelevant dimensions in the lth cluster is $m - m_l$.

A generated synthetic data set is an n-by-m matrix. Its sparsity is defined as

$$\text{Sparsity} = \frac{\text{number of entries with zero-value}}{\text{total number of entries}}. \quad (16)$$

Table 2 gives the algorithm for synthetic data generation.

4.2 Synthetic Data Sets

One hundred synthetic data sets were generated in each type. Each data set has 135 points, 16 dimensions, and nine clusters. The subspace ratio s was set to 0.375, which is equivalent to the average six relevant dimensions in a cluster. The parameter MINMU and MAXMU were set to 0 and 100, respectively, for all data sets.

To study the relationship between the value dispersion of a dimension and its weight value, we generated the following three types of synthetic data by specifying different variances for clusters in each data set:

- *Type I*. All the relevant dimensions in a cluster have equal importance, so we assigned the same variance to them.
- *Type II*. Assuming that some relevant dimensions are more important than others, we assigned small variances to the important relevant dimensions and

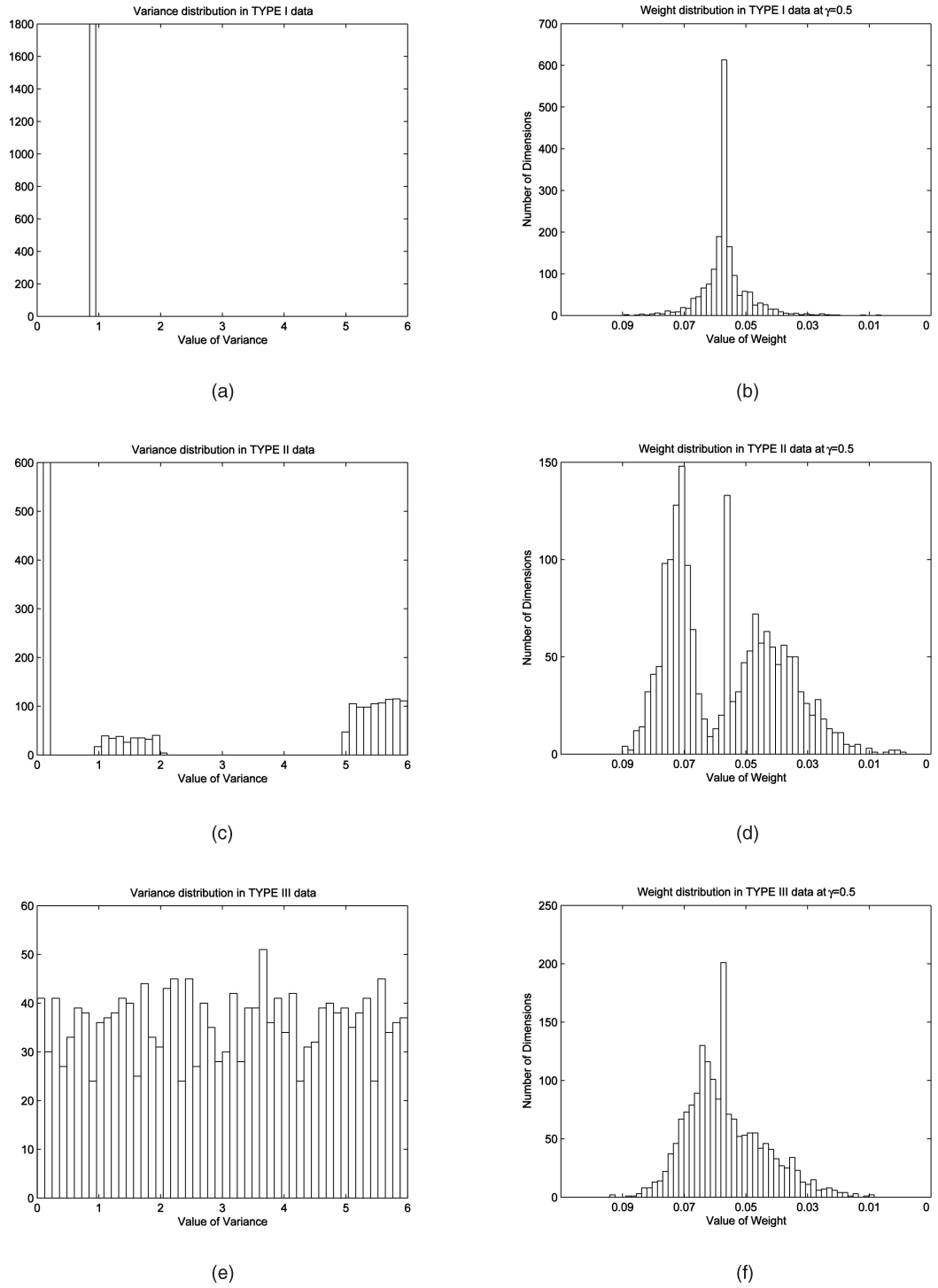

Fig. 2. (a), (c), and (e) show the distributions of dimensions over different variances for three data types, TYPE I, TYPE II, and TYPE III. (b), (d), and (f) are the distributions of dimensions against values of weights. $\gamma = 0.5$.

large variances to the less important relevant dimensions.

- *Type III*. Each relevant dimension is randomly assigned a variance.

For the Type II data, the cluster variances were randomly selected from three ranges [0.1, 0.2], [1, 2], and [5, 6]. For the Type III data, the cluster variances were randomly selected from range [0, 6]. Parameter ϵ was set to 0.01 for generating the random values of irrelevant dimensions and the value range was set to [0, 5].

4.3 Simulation Results

We conducted extensive experiments on the 100 synthetic data sets, investigated the relationship between dimension variances and weight values and the property of parameter γ, and compared the performance of the new algorithm on clustering performance with other subspace clustering algorithms. Some results are reported below.

Fig. 2 shows the relationships between dimension variances and weights in three types of data sets. In the 100 data sets, there were a total of 1,800 relevant dimensions

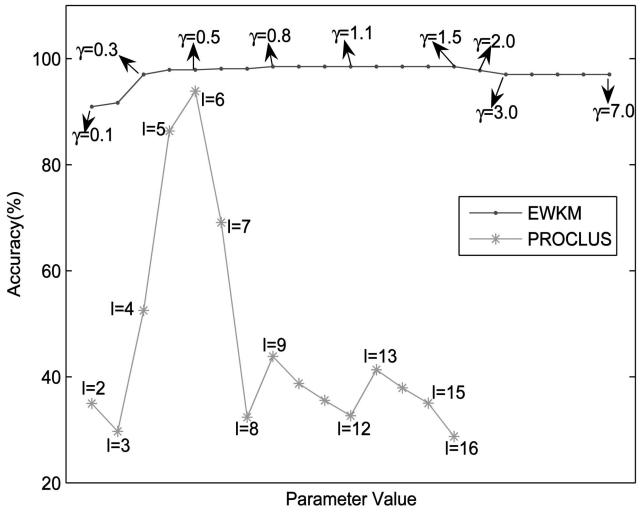

Fig. 3. The clustering accuracy of *EWKM* and *PROCLUS* on the 100 synthetic data sets.

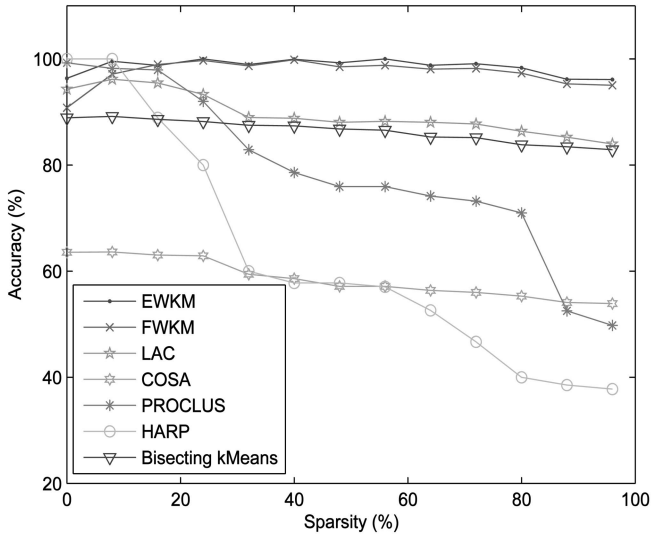

Fig. 4. The clustering accuracy of different algorithms.

in 900 clusters. Figs. 2a, 2c, and 2e are the distributions of relevant dimensions over variance in three types of data sets, whereas Figs. 2b, 2d, and 2f show the distributions of dimensions over weight values. We can see in Fig. 2a that all relevant dimensions had the same variance in the Type I data sets. This type of data resulted from the fact that most dimension weight values were equal or close, as shown in Fig. 2b. This indicates that relevant dimensions with the same distribution would make a similar contribution in identifying clusters in subspaces.

Fig. 2c shows the distribution of dimensions over variance in the Type II data sets. We can observe that the variances for dimensions fall in three ranges [0.1, 0.2], [1, 2], and [5, 6]. Three peaks in the distribution of dimensions over weight values are shown in Fig. 2d. The three peaks correspond to the three variance ranges. This implies that, from the weight values, we are able to relate the weight values to the relevant dimensions in the data sets. Because the Type III data sets randomly selected the variances for dimensions, the distribution of dimensions in Fig. 2f is evenly spread in the range [0, 6]. However, the importance of relevant dimensions is still identifiable from the weight values as shown in Fig. 2f.

To investigate how the parameter γ affects the clustering accuracy, we used the *EWKM* algorithm to cluster the 100 synthetic data sets with different γ values and computed the average accuracy. The clustering accuracy was computed as

$$\text{Clustering Accuracy} = \frac{\sum_{l=1}^{k} D_l}{n},$$

where D_l is the number of points correctly identified in the genuine cluster l, and n is the number of points in the data set.

Fig. 3 shows the average clustering accuracy over 100 data sets with different γ values. We can see that high accuracy was obtained in a large range of γ values [0.3, 7]. These results indicate that the clustering results were not sensitive to the change of γ values, which is a good property of the algorithm. To demonstrate this good property in Fig. 3, we also show the clustering results of the *PROCLUS* [7] over the 100 data sets with different values of parameter l for the average number of relevant dimensions in subspaces. We can see that the clustering results fluctuate as the parameter l was chosen at different values. These results indicate that the clustering results were very sensitive to l, which makes the algorithm difficult to use.

Fig. 4 shows the comparison results of seven clustering algorithms, including *EWKM* and our previous clustering algorithm *FWKM* [22]. Here, *Bisecting k-means* [36] is not a subspace clustering algorithm. *PROCLUS* [7] and *HARP* [11] are two hard subspace clustering algorithms. *LAC* [19] and *COSA* [20] are two other soft subspace clustering algorithms. We can see that *EWKM* outperformed all other algorithms, although *FWKM* is very close. The performances of *LAC* and *COSA* are not affected by the data sparsity. However, we find that their whole clustering qualities are worse than *EWKM* and *FWKM*. The reason is that even though *LAC* and *COSA* deal with sparse problem for high-dimensional data, they adopt an approximation process to minimize their objective functions so that some raw information may be missed. The clustering accuracy of the two hard subspace clustering algorithms *PROCLUS* and *HARP* dropped quickly as the sparsity increased. These results show that *EWKM* was superior in clustering complex data, such as sparse data.

5 Experimental Results on Real-World Data

In this section, we present the experimental results on real-world data. We first show the comparison results of the *EWKM* algorithm and other clustering algorithms on real text data taken from the University of California, Irvine (UCI) Machine Learning Repository.[1] Then, we present a real application to categorize suppliers for a retail company in China. We used *EWKM* to cluster high-dimensional sparse business transaction data to reclassify suppliers based on their business behaviors.

5.1 Text Data

The text data was the publicly available *20-Newsgroups* data. The original text data was first preprocessed to strip the news messages from the e-mail headers and special tags

1. http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html.

TABLE 3
Summary of Text Data Sets

A2 (97.249%)	n_d	B2 (96.373%)	n_d
alt.atheism	100	talk.politics.mideast	100
comp.graphics	100	talk.politics.misc	100
A4 (97.572%)	n_d	**B4 (97.546%)**	n_d
comp.graphics	100	comp.graphics	100
rec.sport.baseball	100	comp.os.ms-windows	100
sci.space	100	rec.autos	100
talk.politics.mideast	100	sci.electronics	100
A4-U (97.259%)	n_d	**B4-U (97.515%)**	n_d
comp.graphics	120	comp.graphics	120
rec.sport.baseball	100	comp.os.ms-windows	100
sci.space	59	rec.autos	59
talk.politics.mideast	20	sci.electronics	20

TABLE 4
Data Sets for Testing the Scalability

D_{1-6}:	(n=15905, k=20)					
	D_1	D_2	D_3	D_4	D_5	D_6
$m =$	500	800	1100	1300	1700	2000
E_{1-4}:	(m=1100, k=20)					
	E_1	E_2	E_3	E_4		
$n =$	2000	4000	8000	15905		
F_{1-5}:	(n=1500, m=500)					
	F_1	F_2	F_3	F_4	F_5	
$k =$	3	5	7	10	12	

TABLE 5
Evaluation Functions

Entropy	$\sum_{l=1}^{k} \frac{n_l}{n} \left(-\frac{1}{\log k} \sum_{j=1}^{k} \frac{n_{jl}}{n_l} \cdot \log \frac{n_{jl}}{n_l} \right)$
FScore	$\sum_{j=1}^{k} \frac{n_j}{n} \cdot \max_{1 \le l \le k} \left\{ \frac{2 \cdot n_{jl}/n_h \cdot n_{jl}/n_l}{n_{jl}/n_j + n_{jl}/n_l} \right\}$
NMI	$\frac{\sum_{j,l} n_{jl} log\left(\frac{n \cdot n_{jl}}{n_j n_l}\right)}{\sqrt{(\sum_j n_j log \frac{n_j}{n})(\sum_l n_l log \frac{n_l}{n})}}$

and eliminate the stop words and stem words to their root forms. Then, the words were sorted on the inverse document frequency (IDF), and some words were removed if the IDF values were too small or too large. The BOW toolkit [37] was used in preprocessing. The word in each document was weighted by the standard $tf \cdot idf$.

Table 3 lists six data sets built from the *20-Newsgroups* data. Each data set contains two or four categories. n_d indicates the number of documents in each category. The percentage gives the sparsity of the data set. Different data sets have different cluster structures. Data sets A2 and A4 contain categories that are semantically different, whereas data sets B2 and B4 contain semantically close categories. Each category was described by a subset of words. (In this case, the documents in each category are one cluster in the data set, and the words are the dimensions.) Obviously, clustering data sets B2 and B4 is more difficult than clustering data sets A2 and A4 because there are more

TABLE 6
Comparisons of Different Algorithms

Data sets	*Bi-kMeans*	*FWKW*	*EWKM*	*LAC*	*PROCLUS*	*HARP*	*COSA*	*SCAD1*
A2	0.2146	0.2057	**0.1667**	0.3776	0.5254	0.5016	0.9999	0.2777
	0.9650	0.9599	**0.9698**	0.9037	0.7190	0.8894	0.5781	0.9490
	0.7857	0.7961	**0.8342**	0.6304	0.2334	0.4984	0.0008	0.7226
B2	0.5294	0.4014	**0.2807**	0.6206	0.8395	0.9562	0.9973	0.5664
	0.8800	0.9043	**0.9449**	0.7981	0.6604	0.6020	0.5413	0.8661
	0.4706	0.6050	**0.7217**	0.4002	0.0789	0.0299	0.0027	0.4260
A4	**0.1919**	0.2509	0.2350	0.5734	0.5548	0.7671	0.9902	0.4214
	0.9376	0.9003	0.9124	0.6721	0.6450	0.5073	0.3152	0.8383
	0.8083	0.7554	0.7693	0.4719	0.2909	0.2023	0.0099	0.5854
B4	0.6195	0.3574	**0.3118**	0.7227	0.7291	0.8933	0.9819	0.5380
	0.7049	0.8631	**0.8919**	0.5816	0.4911	0.3840	0.3621	0.7711
	0.3822	0.6467	**0.6899**	0.3090	0.0791	0.0538	0.0236	0.4174
A4-U	0.2830	0.1513	**0.1194**	0.1431	0.7342	0.8389	0.8768	0.5286
	0.8961	**0.9591**	0.9571	0.9473	0.5239	0.4819	0.4159	0.8719
	0.7126	0.8480	**0.8655**	0.8384	0.1867	0.1688	0.0187	0.5562
B4-U	0.5357	0.2314	**0.2312**	0.4917	0.5758	0.9535	0.8614	0.3591
	0.6586	0.9205	**0.9229**	0.7363	0.5739	0.3364	0.3599	0.8597
	0.3793	0.7385	**0.7510**	0.4968	0.1684	0.0250	0.0300	0.6442

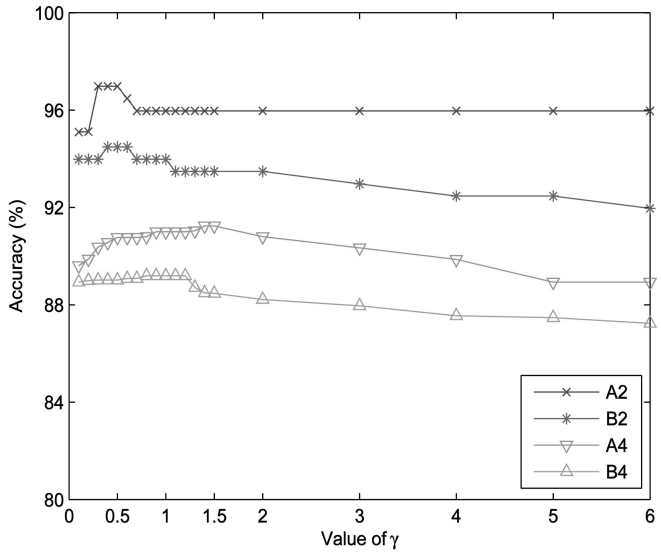

Fig. 5. The effect of γ on clustering accuracy.

overlapping words (dimensions) in data sets B2 and B4. Data sets A4-U and B4-U contain unbalanced documents in each category.

Table 4 lists other 14 data sets used to test the scalability of the algorithm. In the first group of data sets D_{1-6}, each data set contains 15,905 documents in 20 categories. The number of terms (m) in these data sets changes from 500 to 2,000. In the second group of data sets E_{1-4}, the number of categories in each data set was fixed to 20 and the number of terms used to represent these data sets was fixed to 1,100. The number of documents in these data sets changes from 2,000 to 15,905. These two sets of data cover all of the 20 topics in the *20-Newsgroups* collection. In the last group of data sets F_{1-5}, the number of categories k in each data set changes from seven to 12, whereas the number of documents and the number of terms were fixed to 1,500 and 500, respectively. The 12 categories in F_{1-5} are *alt.atheism*, *comp.graphics*, *talk.politics.guns*, *rec.autos*, *soc.religion.christian*, *misc.-forsale*, *sci.crypt*, *comp.sys.ibm.pc.hardware*, *rec.sport.basketball*, *sci.space*, *comp.os.windows*, and *talk.politics.mideast*.

Since the category of each document was known in these data sets, we used the external cluster validation method to evaluate the clustering results by calculating the correspondence between the clusters generated by a clustering algorithm and the inherent categories of the data set. Table 5 lists the three evaluation functions, *Entropy*, *FScore* [38], and the normalized mutual information (*NMI*) [39], which were used to evaluate clustering results. The terms in the three evaluation functions are defined as follows: Assume that a data set with k classes was clustered into k clusters. In these evaluation functions, n_j, n_l are the numbers of documents in class L_j and in cluster C_l, respectively, n_{jl} is the number of documents occurring in both class L_j, and cluster C_l, n is the total number of documents in the data set, and k is the number of clusters equal to the number of classes.

These functions can be interpreted as follows: The smaller the entropy, the better the clustering performance.

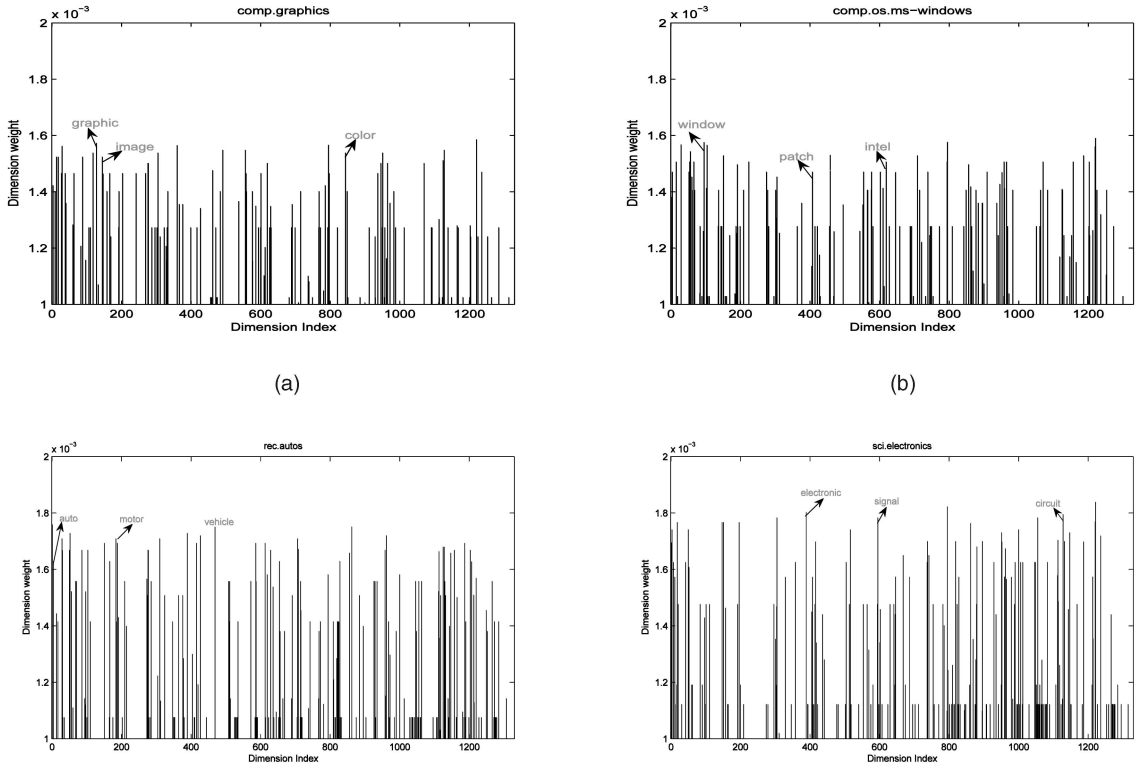

Fig. 6. The weight distributions of keywords in four clusters of data set *B4*. (a) category *comp.graphics*, (b) category *comp.os.ms-windows*, (c) category *rec.autos*, and (d) category *sci.electronics*.

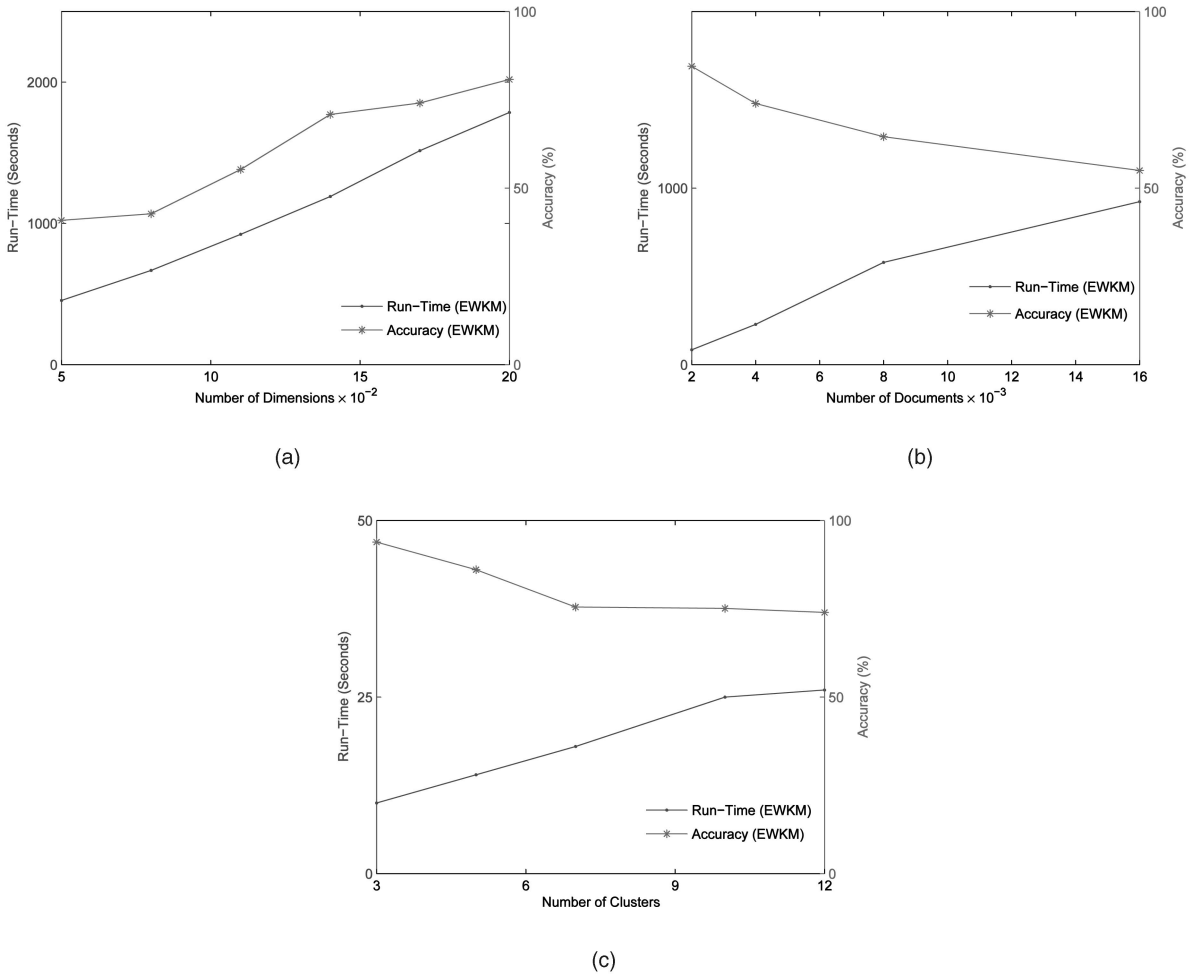

Fig. 7. The relationships between the runtime, clustering accuracy, and different numbers of dimensions/terms, documents, and clusters.

The larger the *FScore/NMI*, the better the clustering performance.

Eight clustering algorithms, *EWKM*, *FWKM* [22], *Bisecting k-means* [36], *LAC* [19], *PROCLUS* [7], *HARP* [11], *COSA* [20], and *SCAD1* [18], were tested on these data sets. The parameter settings for these algorithms were $\gamma = 0.5$ for *EWKM*, $\beta = 1.5$ for *FWKM*, and $S_\xi = 2$ for *SCAD1*. For *PROCLUS*, the parameter l (that is, the average number of relevant dimensions per cluster) was set to 10 percent m, 20 percent m, 30 percent m, 40 percent m, 50 percent m, 60 percent m, 70 percent m, and 80 percent m, respectively, where m is the number of dimensions in the data set. For *LAC*, the parameter τ was set to 0.05, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4, and 12.8, respectively. Except for *COSA* and *HARP*, each clustering algorithm was run 10 times on each parameter value. Here, *LDR* [9] was not tested on these data sets because *LDR* uses the principal components to represent documents, which makes the clustering results hard to explain.

TABLE 7
Summary of Transaction Data

Total number of transactions:	3,945,190
Total number of items:	16,323
Total number of purchased items:	7,441
Number of items categories:	100
Total number of suppliers:	974

Table 6 shows the clustering results on the six text data sets evaluated with the three evaluation measures defined in Table 5. The three figures in each cell represent the values of *Entropy*, *Fscore*, and *NMI*, respectively. The good clustering results are marked in the bold case. We can see that *EWKM* performed better than other algorithms in most cases.

Fig. 5 shows the change of clustering accuracy against different values of γ of *EWKM* on four text data sets in Table 3. We can see that the clustering accuracy was not sensitive to γ when γ changed from 0.05 to 6. These results demonstrate that the clustering result of the *EWKM* algorithm was robust on the parameter γ.

Fig. 6 shows the important relevant dimensions in each cluster for data set *B4*. The x-axis is the dimension index, and the y-axis indicates the dimension weight. From the dimension weights, we can identify several important relevant dimensions that represent important keywords, indicating the topics of the corresponding clusters such as *graphic*, *image*, *color* for category *comp.graphics*; *window*, *patch*, *intel* for category *comp.os.ms-windows*; *auto*, *motor*, *vehicle* for

Cluster	Card.	Relevant Items
2	60	vegetables, eggs, groceries
7	119	deepfrozen chicken, deepfrozen beef, deepfrozen pork, deepfrozen birds, deepfrozen prawn and crab, deepfrozen testacean, deepfrozen fish, deepfrozen lamb and games, deepfrozen vegetables, deepfrozen fruits, deepfrozen foods, deepfrozen seaweed
8	38	peccaries, games, birds
9	34	groceries
10	105	Juice cans, soup cans, vegetables cans, bamboo shoot cans, fruit cans, roe cans, meat cans, fish cans
14	20	snacks, dried foods, instant noodles, instant rice noodles
17	33	breads, dried fruits, snacks, snack seasonings, medicinal materials, fasts, flours, seafood, dried groceries
21	24	eggs, beverage cans, sirup cans
22	28	groceries, dried groceries, snack seasonings
24	20	snacks, dried fruits, instant noodles, snack seasonings, nutlets
26	20	breads, cornstarches, snacks, seasoning powders, sirup cans, cookies, fruit cans, beverages, salads
31	30	powders, icing sugar, rice, seasoning powders, essences, flours, milk powders, additive powders
33	21	medicinal materials, dried mushroom, dried seafood, dried groceries
34	42	seafood, abalones cans, dried abalones, shark fins, birds'nests, seaweeds, abalones, mushroom cans
45	22	sirup cans, sweet beverages, sugars, sugar juices, sugar snacks
49	36	dried Japan food, deepfrozen Japan food

Fig. 8. The number of suppliers and the product categories in the 16 clusters.

category *rec.autos*; and *electronic*, *signal*, *circuit* for category *sci.electronics*. Obviously, such important dimensions were assigned higher weights by the *EWKM* algorithm. We can use these words to visualize the topics of the final clusters.

We used the 14 data sets in Table 4 to test the scalability of *EWKM*. Fig. 7 shows the relationships between the runtime and the numbers of dimensions, documents, and clusters, respectively. We can see that the runtime of *EWKM* increased linearly as the numbers of dimensions (see Fig. 7a), documents (see Fig. 7b), and clusters (see Fig. 7c) increased. These results were consistent with the algorithm analysis in Section 3 and demonstrated that *EWKM* is scalable. Meanwhile, when compared with the existing soft subspace clustering algorithm *COSA* $(O(nmL + hn^2m))$, hard subspace clustering algorithms *LDR* $(O(nm^2k))$ and *HARP*, *EWKM* can satisfy the computational and representational requirements of real-world text data and scale efficiently with increasing data size.

5.2 Business Transactions Data

The objective of this analysis was to help a food retail company in China to categorize its suppliers according to suppliers' business behaviors. Supplier categorization refers to the process of dividing suppliers of an organization into different groups according to the characteristics of the suppliers so that each group of suppliers can be managed differently within the organization. Supplier categorization is an important step in SRM for creating better supplier management strategies to reduce the product sourcing risk and costs and improve business performance.

A supplier's business behavior in a given time period is recorded in business transactions made at different time points. A business transaction records a purchase of food products. The most relevant data attributes in a transaction include the amount, quantity, and price of each product item and the time that the transaction was committed. In this experiment, we analyzed business transactions of 974 suppliers in a 10 and 1/2 months period (from 1 January 2004 to 16 November 2004). The summary of the data is given in Table 7.

The transaction data was converted to the behavior representation in a matrix with 100 columns and 974 rows. Each column represents one item category, and each row represents one supplier. Each entry in the matrix represents the purchase amount of products in a category paid to a supplier. Since suppliers usually can supply products in a few categories, this data matrix was very sparse. The sparsity measured by (16) was 96.760 percent.

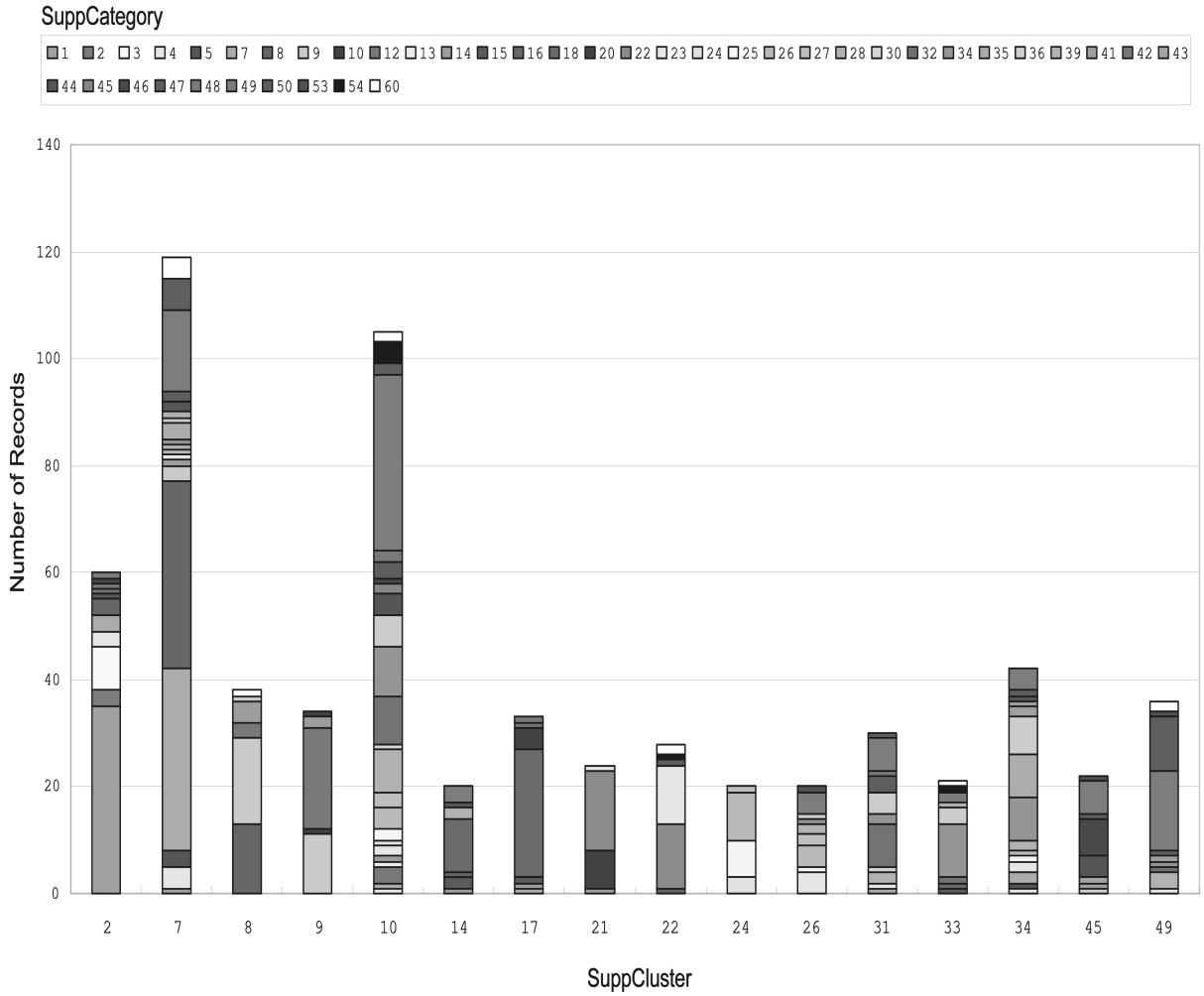

Fig. 9. Comparison of existing supplier categories with the 16 clusters. We can see that suppliers in the same cluster are often divided into more than one category in the existing categorization.

We applied *EWKM* to the behavior matrix to cluster the 974 suppliers into 60 clusters. This was because the company already classified its suppliers into 60 groups based on suppliers' location and the product categories that suppliers can provide. However, suppliers' business behaviors were not considered in the classification. Our result was used to readjust the existing categorization for better selection of suppliers in sourcing.

Fig. 8 shows the characteristics of 16 clusters, each with more than 20 suppliers. The first column is the cluster index, the second is the number of suppliers, and the third gives the set of product categories, which the suppliers in each cluster supplied. The set of product categories in each cluster was identified by the weight values. The product categories in each cluster show the real orders the company made to these suppliers in that cluster in the past. They are very useful for the sourcing staff to select suitable suppliers in purchase orders. For example, the suppliers in cluster 9 may also provide other product categories in their product catalogs. However, according to the company's purchasing history, only groceries were ordered from these suppliers. If the sourcing task was to purchase groceries only, the suppliers in cluster 9 could be selected. If the task was to purchase groceries and vegetables, the suppliers in cluster 2 could be selected. Therefore, these clustering results can provide actionable information for the sourcing staff to make purchase decisions.

The clustering result was compared with the existing supplier categories. Fig. 9 shows the relationships between the generated 16 clusters and the existing categories. From the chart, we can see that the existing supplier categorization is not consistent with the supplier clusters that were generated from the suppliers' business behavior data. From the clustering result, we know that suppliers in the same cluster supplied the same set of product categories to the company. They should be treated as the same category for easy supplier selection. However, they are divided into different categories in the existing categorization. For instance, suppliers in clusters 21 are divided into categories 14 and 20. Using the clustering result, the suppliers in the two categories can be associated by cluster 21. When making purchase orders on the product categories identified in cluster 21, the suppliers in categories 14 and 20 can be selected.

In comparison to the company's existing supplier categorization, the supplier categorization generated from supplier business behavior data has two merits. First, it contains the buying patterns, which can be used to manage

suppliers to enlarge company's profits. The key rule of the supplier categorization is the product categories, but the main product categories provided by a supplier may not be important to the company. Also, the importance of a supplier to the company and the importance of the company to a supplier are all buried in the buying patterns. Corresponding to the clustering results, this work can be done smoothly. Second, we can see from the clustering results shown in the previously that the product categories information is also included in categorization.

6 Conclusions

In this paper, we have presented *EWKM*, a new k-means type subspace clustering algorithm for high-dimensional sparse data. In this algorithm, we simultaneously minimize the within cluster dispersion and maximize the negative weight entropy in the clustering process. Because this clustering process awards more dimensions to make contributions to identification of each cluster, the problem of identifying clusters by few sparse dimensions can be avoided. As such, the sparsity problem of high-dimensional data is tackled. The experimental results on both synthetic and real data sets have shown that the new algorithm outperformed other k-means type algorithms, for example, *Bisecting k-means* and *FWKM*, and subspace clustering methods, for example, *PROCLUS*, *HARP*, *LAC*, *SCAD1*, and *COSA*, in recovering clusters. Except for clustering accuracy, the new algorithm is scalable to large high-dimensional data and easy to use because the input parameter γ is not sensitive. The weight values generated in the clustering process are also useful for other purposes, for instance, identifying the keywords to represent the semantics of text clustering results.

Acknowledgments

The authors would like to thank Kevin Y. Yip for providing the useful online projected clustering platform *BioSphere*[2] and suggestions on experiments. M.K. Ng's research was suppoted in part by RGC grants 7046/03P, 7035/04P, 7035/05P, 7117/05E, and HKBU FRGs. This research was also supported by China National 863 project 2006AA01A124.

References

[1] X. Zhang, J.Z. Huang, D. Qian, J. Xu, and L. Jing, "Supplier Categorization with k-Means Type Subspace Clustering," *Proc. Eighth Asia Pacific Web Conf.*, 2006.

[2] A.K. Jain, M.N. Murty, and P.L. Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264-323, 1999.

[3] M. Steinbach, L. Ertoz, and V. Kumar, *The Challenges of Clustering High Dimensional Data,* http://www-users.cs.umn.edu/~ertoz/papers/clustering_chapter.pdf, 2003.

[4] D. Cai, X. He, and J. Han, "Document Clustering Using Locality Preserving Indexing," *IEEE Trans. Knowledge and Data Eng.*, vol. 17, no. 12, Dec. 2005.

[5] D.R. Swanson, "Medical Literature as a Potential Source of New Knowledge," *Bull. Medical Library Assoc.*, vol. 78, no. 1, Jan. 1990.

2. http://bio1.no-ip.com:8299/BiosphereServer/.

[6] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 94-105, 1998.

[7] C. Aggarwal, C. Procopiuc, J.L. Wolf, P.S. Yu, and J.S. Park, "Fast Algorithms for Projected Clustering," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 61-72, 1999.

[8] C.C. Aggarwal and P.S. Yu, "Finding Generalized Projected Clusters in High Dimensional Spaces," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 70-81, 2000.

[9] K. Chakrabarti and S. Mehrotra, "Local Dimensionality Reduction: A New Approach to Indexing High Dimensional Spaces," *Proc. 26th Int'l Conf. Very Large Data Bases,* pp. 89-100, 2000.

[10] C.M. Procopiuc, M. Jones, P.K. Agarwal, and T.M. Murali, "A Monte Carlo Algorithm for Fast Projective Clustering," *Proc. ACM SIGMOD Int'l Conf. Management of Data,* pp. 418-427, 2002.

[11] K.Y. Yip, D.W. Cheung, and M.K. Ng, "A Practical Projected Clustering Algorithm," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 11, pp. 1387-1397, Nov. 2004.

[12] K.Y. Yip, D.W. Cheung, and M.K. Ng, "On Discovery of Extremely Low-Dimensional Clusters Using Semi-Supervised Projected Clustering," *Proc. 21st Int'l Conf. Data Eng.,* pp. 329-340, 2005.

[13] W.S. Desarbo, J.D. Carroll, L.A. Clark, and P.E. Green, "Synthesized Clustering: A Method for Amalgamating Clustering Bases with Differential Weighting Variables," *Psychometrika,* vol. 49, pp. 57-78, 1984.

[14] G.W. Milligan, "A Validation Study of a Variable Weighting Algorithm for Cluster Analysis," *J. Classification,* vol. 6, pp. 53-71, 1989.

[15] D.S. Modha and W.S. Spangler, "Feature Weighting in k-Means Clustering," *Machine Learning,* vol. 52, pp. 217-237, 2003.

[16] Y. Chan, W. Ching, M.K. Ng, and J.Z. Huang, "An Optimization Algorithm for Clustering Using Weighted Dissimilarity Measures," *Pattern Recognition,* vol. 37, no. 5, pp. 943-952, 2004.

[17] H. Frigui and O. Nasraoui, "Unsupervised Learning of Prototypes and Attribute Weights," *Pattern Recognition,* vol. 37, no. 3, pp. 567-581, 2004.

[18] H. Frigui and O. Nasraoui, "Simultaneous Clustering and Dynamic Keyword Weighting for Text Documents," *Survey of Text Mining,* Michael Berry, ed., pp. 45-70, Springer, 2004.

[19] C. Domeniconi, D. Papadopoulos, D. Gunopulos, and S. Ma, "Subspace Clustering of High Dimensional Data," *Proc. SIAM Int'l Conf. Data Mining,* 2004.

[20] J.H. Friedman and J.J. Meulman, "Clustering Objects on Subsets of Attributes," *J. Royal Statistical Soc. B,* vol. 66, no. 4, pp. 815-849, 2004.

[21] J.Z. Huang, M.K. Ng, H. Rong, and Z. Li, "Automated Variable Weighting in k-Means Type Clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 5, pp. 1-12, May 2005.

[22] L. Jing, M.K. Ng, J. Xu, and J.Z. Huang, "Subspace Clustering of Text Documents with Feature Weighting k-Means Algorithm," *Proc. Ninth Pacific-Asia Conf. Knowledge Discovery and Data Mining,* pp. 802-812, 2005.

[23] L. Parsons, E. Haque, and H. Liu, "Subspace Clustering for High Dimensional Data: A Review," *SIGKDD Explorations,* vol. 6, no. 1, pp. 90-105, 2004.

[24] C.H. Cheng, A.W. Fu, and Y. Zhang, "Entropy-Based Subspace Clustering for Mining Numerical Data," *Proc. Fifth ACM SIGKDD Int'l Conf. Knowledge and Data Mining,* pp. 84-93, 1999.

[25] S. Goil, H. Nagesh, and A. Choudhary, "Mafia: Efficient and Scalable Subspace Clustering for Very Large Data Sets," Technical Report CPDC-TR-9906-010, Northwest Univ., 1999.

[26] R.T. Ng and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining," *Proc. 20th Int'l Conf. Very Large Data Bases,* pp. 144-155, Sept. 1994.

[27] K.G. Woo and J.H. Lee, "Findit: A Fast and Intelligent Subspace Clustering Algorithm Using Dimension Voting," PhD dissertation, Korea Advanced Inst. of Science and Technology, 2002.

[28] J. Yang, W. Wang, H. Wang, and P. Yu, "δ-Clusters: Capturing Subspace Correlation in a Large Data Set," *Proc. 18th Int'l Conf. Data Eng.,* pp. 517-528, 2002.

[29] E. Keogh, K. Chakrabarti, S. Mehrotra, and M. Pazzani, "Local Adaptive Dimensionality Reduction for Indexing Large Time Series Databases," *Proc. ACM SIGMOD Int"l Conf. Management of Data,* 2001.

[30] G. De Soete, "Optimal Variable Weighting for Ultrametric and Additive Tree Clustering," *Quality and Quantity,* vol. 20, pp. 169-180, 1986.
[31] G. De Soete, "OVWTRE: A Program for Optimal Variable Weighting for Ultrametric and Additive Tree Fitting," *J. Classification,* vol. 5, pp. 101-104, 1988.
[32] V. Makarenkov and P. Legendre, "Optimal Variable Weighting for Ultrametric and Additive Trees and k-Means Partitioning: Methods and Software," *J. Classification,* vol. 18, pp. 245-271, 2001.
[33] C. Domeniconi, "Locally Adaptive Techniques for Pattern Classification," PhD dissertation, 2002.
[34] L. Jing, M.K. Ng, J. Xu, and J.Z. Huang, "On the Performance of Feature Weighting k-Means for Text Subspace Clustering," *Proc. Sixth Int'l Conf. Web-Age Information Management,* pp. 502-512, 2005.
[35] M. Zait and H. Messatfa, "A Comparative Study of Clustering Methods," *Future Generation Computer Systems,* vol. 13, pp. 149-159, 1997.
[36] M. Steinbach, G. Karypis, and V. Kumar, "A Comparison of Document Clustering Techniques," *Proc. KDD Workshop Text Mining,* 2000.
[37] A.K. McCallum, "Bow: A Toolkit for Statistical Language Modeling, Text Retrieval, Classification and Clustering," http://www.cs.cmu.edu/mccallum/bow, 1996.
[38] Y. Zhao and G. Karypis, "Comparison of Agglomerative and Partitional Document Clustering Algorithms," Technical Report #02-014, Univ. of Minnesota, 2002.
[39] S. Zhong and J. Ghosh, "A Comparative Study of Generative Models for Document Clustering," *Proc. SDW Workshop Clustering High-Dimensional Data and Its Applications,* May 2003.

Liping Jing received the BSc and MPhil degrees in computer science from the Northern Jiaotong University in 2000 and 2003, respectively. She is a PhD student in the Department of Mathematics and E-Business Technology Institute at the University of Hong Kong. Her research interests are in the areas of data mining, subspace clustering algorithm, ontology-based programming, and business intelligence.

Michael K. Ng is a professor in the Department of Mathematics, Hong Kong Baptist University. As an applied mathematician, his main research areas include bioinformatics, data mining, operations research, and scientific computing. He has published and edited five books and published more than 140 journal papers. He is the principal editor of the *Journal of Computational and Applied Mathematics* and the associate editor of the *SIAM Journal on Scientific Computing*, *Numerical Linear Algebra with Applications*, *International Journal of Data Mining and Bioinformatics*, and *Multidimensional Systems and Signal Processing*.

Joshua Zhexue Huang received the PhD degree from the Royal Institute of Technology in Sweden. He is the assistant director of E-Business Technology Institute (ETI) at the University of Hong Kong. Before joining ETI in 2000, he was a senior consultant at the Management Information Principles, Australia, consulting on data mining and business intelligence systems. From 1994 to 1998, he was a research scientist at Commonwealth Scientific and Industrial Research Organisation (CSIRO) in Australia. His research interests include data mining, machine learning, clustering algorithms, and grid computing.