Big Data and Information Analytics
©American Institute of Mathematical Sciences
Volume **1**, Number **1**, January **2016**
doi:10.3934/bdia.2016.1.93
pp. **93–109**

A SOFT SUBSPACE CLUSTERING ALGORITHM WITH LOG-TRANSFORMED DISTANCES

Guojun Gan

Department of Mathematics
University of Connecticut
196 Auditorium Rd, Storrs, CT 06269-3009, USA

Kun Chen

Department of Statistics
University of Connecticut
215 Glenbrook Road, Storrs, CT 06269, USA

Abstract. Entropy weighting used in some soft subspace clustering algorithms is sensitive to the scaling parameter. In this paper, we propose a novel soft subspace clustering algorithm by using log-transformed distances in the objective function. The proposed algorithm allows users to choose a value of the scaling parameter easily because the entropy weighting in the proposed algorithm is less sensitive to the scaling parameter. In addition, the proposed algorithm is less sensitive to noises because a point far away from its cluster center is given a small weight in the cluster center calculation. Experiments on both synthetic datasets and real datasets are used to demonstrate the performance of the proposed algorithm.

1. **Introduction.** In data clustering or cluster analysis, the goal is to divide a set of objects into homogeneous groups called clusters [10, 18, 20, 26, 12, 1]. For high-dimensional data, clusters are usually formed in subspaces of the original data space and different clusters may relate to different subspaces. To recover clusters embedded in subspaces, subspace clustering algorithms have been developed, see for example [2, 15, 19, 17, 9, 21, 16, 22, 3, 25, 7, 11, 13]. Subspace clustering algorithms can be classified into two categories: hard subspace clustering algorithms and soft subspace clustering algorithms.

In hard subspace clustering algorithms, the subspaces in which clusters embed are determined exactly. In other words, each attribute of the data is either associated with a cluster or not associated with the cluster. For example, the subspace clustering algorithms developed in [2] and [15] are hard subspace clustering algorithms. In soft subspace clustering algorithms, the subspaces of clusters are not determined exactly. Each attribute is associated to a cluster with some probability. If an attribute is important to the formation of a cluster, then the attribute is associated to the cluster with high probability. Examples of soft subspace clustering algorithms include [19], [9], [21], [16], and [13].

In soft subspace clustering algorithms, the attribute weights associated with clusters are automatically determined. In general, the weight of an attribute for a cluster is inversely proportional to the dispersion of the attribute in the cluster. If

2010 *Mathematics Subject Classification.* Primary: 62H30, 68T10, 91C20; Secondary: 62P10.
Key words and phrases. Data clustering, subspace clustering, attribute weighting, k-means.

the values of an attribute in a cluster is relatively compact, then the attribute will be assigned a relatively high value. In the FSC algorithm [16], for example, the attribute weights are calculated as

$$w_{lj} = \frac{1}{\sum_{h=1}^{d} \left(\frac{V_{lj}+\epsilon}{V_{lh}+\epsilon}\right)^{\frac{1}{\alpha-1}}}, \quad l = 1, 2, \ldots, k,\ j = 1, 2, \ldots, d, \tag{1}$$

where ϵ is a small positive number used to prevent dividing by zero, $\alpha > 1$ is a parameter used to control the smoothness of the attribute weights, and

$$V_{lj} = \sum_{\mathbf{x} \in C_l} (x_j - z_{lj})^2. \tag{2}$$

Here k is the number of clusters, d is the number of attributes, and $\mathbf{z}_l$ is the center of the lth cluster C_l. In the EWKM algorithm [21], the attribute weights are calculated as

$$w_{lj} = \frac{\exp\left(-\frac{V_{lj}}{\gamma}\right)}{\sum_{s=1}^{d} \exp\left(-\frac{V_{ls}}{\gamma}\right)}, \quad k = 1, 2, \ldots, n,\ l = 1, 2, \ldots, d, \tag{3}$$

where $\gamma > 0$ is a parameter used to control the smoothness of the attribute weights.

One drawback of the FSC algorithm is that a positive value of ϵ is required in order to prevent dividing by zero when an attribute has identical values in a cluster. Using the entropy weighting, the EWKM algorithm does not suffer from the problem of dividing by zero. However, the attribute weights calculated in the EWKM algorithm are sensitive to the parameter γ when the range of the attribute dispersions (e.g., V_{lj}) in a cluster is large. For example, suppose that a dataset has two attributes, whose dispersions in a cluster are 10 and 30, respectively. If we use a small value of γ such as $\gamma = 1$, the attribute weights will be

$$w_1 = \frac{e^{-10}}{e^{-10} + e^{-30}} = \frac{1}{1 + e^{-20}} = 1, \quad w_2 = \frac{e^{-30}}{e^{-10} + e^{-30}} = \frac{1}{1 + e^{20}} = 0.$$

If we use $\gamma = 10$, the attribute weights will be

$$w_1 = \frac{e^{-1}}{e^{-1} + e^{-3}} = \frac{1}{1 + e^{-2}} = 0.88, \quad w_2 = \frac{e^{-3}}{e^{-1} + e^{-3}} = \frac{1}{1 + e^{2}} = 0.12.$$

From the above example we see that choosing an appropriate value for the parameter γ is a difficult task when the attribute dispersions in a cluster is large. Feature group weighting has been introduced to address the issue [7, 14].

In this paper, we address the issue from a different perspective. Unlike the group feature weighting approach, the approach we employ in this paper involves using the log transformation to transform the distances so that the attribute weights are not dominated by a single attribute with the smallest dispersion. In particular, we present a soft subspace clustering algorithm called the LEKM algorithm (log-transformed entropy weighting k-means) to address the aforementioned problem. The LEKM algorithm extends the EWKM algorithm by using log-transformed distances in its objective function. The resulting attribute dispersions in a cluster are more compact than those from the EWKM algorithm. Due to the small difference of the attribute dispersions, the LEKM algorithm is less sensitive to the parameter than other soft subspace clustering algorithms are.

The remaining part of this paper is structured as follows. In Section 2, we give a brief review of the LAC algorithm [9] and the EWKM algorithm [21]. In Section 3, we present the LEKM algorithm in detail. In Section 4, we present numerical

experiments to demonstrate the performance of the LEKM algorithm. Section 5 concludes the paper with some remarks.

2. **Related work.** In this section, we introduce the EWKM algorithm [21] and the LAC algorithm [9], which are soft subspace clustering algorithms using the entropy weighting.

2.1. **The EWKM algorithm.** Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ be n data points, each of which is described by d attributes. Let k be the desired number of clusters. Then the objective function of the EWKM algorithm is defined as follows [21]:

$$F(U, W, Z) = \sum_{l=1}^{k} \left[\sum_{i=1}^{n} \sum_{j=1}^{d} u_{il} w_{lj} (x_{ij} - z_{lj})^2 + \gamma \sum_{j=1}^{d} w_{lj} \ln w_{lj} \right], \tag{4}$$

where $\gamma > 0$ is a parameter, $U = (u_{il})_{n \times k}$ is a $n \times k$ partition matrix, and $W = (w_{lj})_{k \times d}$ is a $k \times d$ weight matrix. In addition, the partition matrix U and the weight matrix W satisfy the following conditions:

$$\sum_{l=1}^{k} u_{il} = 1, \quad i = 1, 2, \ldots, n, \tag{5a}$$

$$u_{il} \in \{0, 1\}, \quad i = 1, 2, \ldots, n, \; l = 1, 2, \ldots, k, \tag{5b}$$

$$\sum_{j=1}^{d} w_{lj} = 1, \quad l = 1, 2, \ldots, k, \tag{5c}$$

and

$$w_{lj} > 0, \quad l = 1, 2, \ldots, k, \; j = 1, 2, \ldots, d. \tag{5d}$$

Like the k-means algorithm [23, 4], the EWKM algorithm tries to minimize the objective function using an iterative process. At the beginning, the EWKM algorithm initializes the cluster centers by selecting k points from the dataset randomly and initializes the attribute weights with equal values. Then the EWKM algorithm keeps updating U, W, and Z one at a time by fixing the other two. Given W and Z, the partition matrix U is updated as

$$u_{il} = \begin{cases} 1, & \text{if } \sum_{j=1}^{d} w_{lj} (x_{ij} - z_{lj})^2 \leq \sum_{j=1}^{d} u_{is} w_{sj} (x_{ij} - z_{sj})^2 \text{ for } 1 \leq s \leq k, \\ 0, & \text{if otherwise,} \end{cases}$$

for $i = 1, 2, \ldots, n$ and $l = 1, 2, \ldots, k$. Given U and Z, the weight matrix W is updated as

$$w_{lj} = \frac{\exp\left(-\frac{V_{lj}}{\gamma}\right)}{\sum_{s=1}^{d} \exp\left(-\frac{V_{ls}}{\gamma}\right)}$$

for $l = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, d$, where

$$V_{lj} = \sum_{i=1}^{n} u_{il} (x_{ij} - z_{lj})^2.$$

Given U and W, the cluster centers are updated as

$$z_{lj} = \frac{\sum_{i=1}^{n} u_{il} x_{ij}}{\sum_{i=1}^{n} u_{il}}$$

for $l = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, d$. The runtime complexity of one iteration of the EWKM algorithm is $O(nkd)$.

The parameter γ in the EWKM algorithm is used to control the smoothness of the attribute weights. If γ approaches to infinity, then all attributes have the same weights. In such cases, the EWKM algorithm becomes the standard k-means algorithm. Since the attribute weights are based on exponential normalization, the weights are sensitive to the parameter γ when the attribute dispersions (e.g., V_{lj}) have a wide range.

2.2. **The LAC algorithm.** The LAC algorithm (Locally Adaptive Clustering) [9] and the EWKM algorithm are similar soft subspace clustering algorithms in that both algorithms discover subspace clusters via exponential weighting of attributes. However, the LAC algorithm differs from the EWKM algorithm in the definition of objective function. Clusters found by the LAC algorithm are referred to as weighted clusters. The objective function of the LAC algorithm is defined as

$$E(\mathcal{C}, Z, W) = \sum_{l=1}^{k} \sum_{j=1}^{d} \left(w_{lj} \frac{1}{|C_l|} \sum_{\mathbf{x} \in C_l} (x_j - z_{lj})^2 + h w_{lj} \log w_{lj} \right), \tag{6}$$

where k is the number of clusters, d is the number of attributes, $Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}$ is a set of cluster centers, $W = (w_{lj})_{k \times d}$ is a weight matrix, $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ is a set of clusters, and $h > 0$ is a parameter. The weight matrix also satisfies the conditions given in Equations (5c) and (5d).

Like the k-means algorithm and the EWKM algorithm, the LAC algorithm also employs an iterative process to optimize the objective function. Similar to the EWKM algorithm, the LAC algorithm initializes the cluster centers by selecting k points from the dataset randomly and initializes the attribute weights with equal values. Given the set of cluster centers Z and the set of weight vectors W, the clusters are determined as follows:

$$S_l = \left\{ \mathbf{x} : \sum_{j=1}^{d} w_{lj}(x_j - z_{lj})^2 < \sum_{j=1}^{d} w_{sj}(x_j - z_{sj})^2, \forall s \neq l \right\} \tag{7}$$

for $l = 1, 2, \ldots, k$. Given the set of cluster centers Z and the set of clusters $\{S_1, S_2, \ldots, S_k\}$, the set of weight vector is determined as follows:

$$w_{lj} = \frac{\exp(-V_{lj})/h}{\sum_{s=1}^{d} \exp(-V_{ls}/h)} \tag{8}$$

for $l = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, d$, where

$$V_{lj} = \frac{1}{|S_l|} \sum_{\mathbf{x} \in S_l} (x_j - z_{lj})^2.$$

Given the set of clusters $\{S_1, S_2, \ldots, S_k\}$, the cluster centers are updated as follows:

$$z_{lj} = \frac{1}{|S_l|} \sum_{\mathbf{x} \in S_l} x_j \tag{9}$$

for $l = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, d$. The runtime complexity of one iteration of the LAC algorithm is $O(nkd)$.

Comparing Equation (6) with Equation (4), we see that the distances in the objective function of the LAC algorithm are normalized by the sizes of the corresponding clusters. As a result, the dispersions (i.e., V_{lj}) calculated in the LAC algorithm are smaller than those calculated in the EWKM algorithm. However,

the dispersions calculated in the LAC algorithm can still have a wide range for small-sample high-dimensional data such as gene expression data [8].

3. **The LEKM algorithm.** In this section, we present the LEKM algorithm. The LEKM algorithm is similar to the EWKM algorithm [21] and the LAC algorithm [9] in that the entropy weighting is used to determine the attribute weights.

Let $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be a dataset containing n points, each of which is described by d numerical features or attributes. Let $Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}$ be a set of cluster centers, where k is the number of clusters. Then the objective function of the LEKM algorithm is defined as

$$\begin{aligned} & P(U, W, Z) \\ = & \sum_{l=1}^{k}\sum_{i=1}^{n} u_{il} \sum_{j=1}^{d} w_{lj} \ln\left[1 + (x_{ij} - z_{lj})^2\right] + \lambda \sum_{l=1}^{k}\sum_{i=1}^{n} u_{il} \sum_{j=1}^{d} w_{lj} \ln w_{lj} \\ = & \sum_{l=1}^{k}\sum_{i=1}^{n} u_{il} \left[\sum_{j=1}^{d} w_{lj} \ln\left[1 + (x_{ij} - z_{lj})^2\right] + \lambda \sum_{j=1}^{d} w_{lj} \ln w_{lj}\right], \end{aligned} \tag{10}$$

where $U = (u_{il})_{n\times k}$ is a $n \times k$ binary matrix satisfying Equations (5a) and (5b), $W = (w_{lj})_{k\times d}$ is a $k\times d$ satisfying Equations (5c) and (5d), and $\lambda > 0$ is a parameter. In the above equation, x_{ij} and z_{lj} denote the values of $\mathbf{x}_i$ and $\mathbf{z}_l$ in the jth attribute, respectively. The matrix U is the partition matrix in the following sense. If $u_{il} = 1$, then the point $\mathbf{x}_i$ belongs to the lth cluster. The matrix W is the weight matrix containing the attribute weights. If w_{lj} is relatively large, then the jth attribute is important for the formulation of the lth cluster.

Similar to the EWKM algorithm, the LEKM algorithm tries to minimize the objective function given in Equation (10) iteratively by finding the optimal value of U, W, and Z according to the following theorems.

Theorem 3.1. *Let W and Z be fixed. Then the partition matrix U that minimizes the objective function $P(U, W, Z)$ is given by*

$$u_{il} = \begin{cases} 1, & \text{if } D(\mathbf{x}_i, \mathbf{z}_l) \le D(\mathbf{x}_i, \mathbf{z}_s) \text{ for all } s = 1, 2, \ldots, k; \\ 0, & \text{if otherwise,} \end{cases} \tag{11}$$

for $i = 1, 2, \ldots, n$ and $l = 1, 2, \ldots, k$, where

$$D(\mathbf{x}_i, \mathbf{z}_s) = \sum_{j=1}^{d} w_{lj} \ln\left[1 + (x_{ij} - z_{sj})^2\right] + \lambda \sum_{j=1}^{d} w_{lj} \ln w_{lj}.$$

Proof. Since W and Z are fixed and the rows of the partition matrix U are independent of each other, the objective function is minimized if for each $i = 1, 2, \ldots, n$, the following function

$$f(u_{i1}, u_{i2}, \ldots, u_{ik}) = \sum_{l=1}^{k} u_{il} D(\mathbf{x}_i, \mathbf{z}_l) \tag{12}$$

is minimized. Note that $u_{il} \in \{0, 1\}$ and

$$\sum_{l=1}^{k} u_{il} = 1.$$

The function defined in Equation (12) is minimized if Equation (11) holds. This completes the proof. □

Theorem 3.2. *Let U and Z be fixed. Then the weight matrix W that minimizes the objective function $P(U, W, Z)$ is given by*

$$w_{lj} = \frac{\exp\left(-\frac{V_{lj}}{\lambda}\right)}{\sum_{s=1}^{d} \exp\left(-\frac{V_{ls}}{\lambda}\right)} \tag{13}$$

for $l = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, d$, where

$$V_{lj} = \frac{\sum_{i=1}^{n} u_{il} \ln\left[1 + (x_{ij} - z_{lj})^2\right]}{\sum_{i=1}^{n} u_{il}}.$$

Proof. The weight matrix W that minimizes the objective function $P(U, W, Z)$ subject to

$$\sum_{j=1}^{d} w_{lj} = 1, \quad l = 1, 2, \ldots, k,$$

is the matrix W that minimizes the following function

$$\begin{aligned} f(W) &= P(U, W, Z) + \sum_{l=1}^{k} \beta_l \left(\sum_{j=1}^{d} w_{lj} - 1 \right) \\ &= \sum_{l=1}^{k} \sum_{i=1}^{n} u_{il} \left[\sum_{j=1}^{d} w_{lj} \ln\left[1 + (x_{ij} - z_{lj})^2\right] + \lambda \sum_{j=1}^{d} w_{lj} \ln w_{lj} \right] \\ &\quad + \sum_{l=1}^{k} \beta_l \left(\sum_{j=1}^{d} w_{lj} - 1 \right). \end{aligned} \tag{14}$$

The weight matrix W that minimizes Equation (14) satisfies the following equations

$$\frac{\partial f(W)}{\partial w_{lj}} = \sum_{i=1}^{n} u_{il} \left(\ln\left[1 + (x_{ij} - z_{lj})^2\right] + \lambda \ln w_{lj} + \lambda \right) + \beta_l = 0$$

for $l = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, d$, and

$$\frac{\partial f(W)}{\partial \beta_l} = \sum_{j=1}^{d} w_{lj} - 1 = 0$$

for $l = 1, 2, \ldots, k$. Solving the above equations leads to Equation (13). □

From Equation (13) we see that the attribute weights of the lth cluster are the exponential normalizations of V_{l1}, V_{l2}, ..., V_{ld}. Since V_{lj} is the sum of log-transformed distances, the range of the magnitudes of V_{l1}, V_{l2}, ..., V_{ld} is small. Hence the weights are less sensitive to the parameter λ.

Theorem 3.3. *Let U and W be fixed. Then the set of cluster centers Z that minimizes the objective function $P(U, W, Z)$ satisfies the following nonlinear equations*

$$z_{lj} = \frac{\sum_{i=1}^{n} u_{il} \left[1 + (x_{ij} - z_{lj})^2\right]^{-1} x_{ij}}{\sum_{i=1}^{n} u_{il} \left[1 + (x_{ij} - z_{lj})^2\right]^{-1}} \tag{15}$$

for $l = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, d$.

Proof. If the set of cluster centers Z minimizes the objective function $P(U, W, Z)$, then for all $l = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, d$, the derivative of $P(U, W, Z)$ with respect to w_{lj} is equal to zeros. In other words, we have

$$\frac{\partial P}{\partial z_{lj}} = w_{lj} \sum_{i=1}^{n} u_{il} \left[1 + (x_{ij} - z_{lj})^2\right]^{-1} [-2(x_{ij} - z_{lj})] = 0.$$

Since $w_{lj} > 0$, we have

$$\sum_{i=1}^{n} u_{il} \left[1 + (x_{ij} - z_{lj})^2\right]^{-1} [-2(x_{ij} - z_{lj})] = 0,$$

from which Equation (15) follows. □

In the standard k-means algorithm, the EWKM algorithm, and the LAC algorithm, the center of a cluster is calculated as the average of the points in the cluster. In the LEKM algorithm, however, the center of a cluster is governed by a nonlinear equation in such a way that the center is a weighted average of the points in the cluster. In addition, if a point is far away from its center, then the point is given a low weight in the center calculation. As a result, the LEKM algorithm is less sensitive to outliers than the EWKM algorithm and the LAC algorithm. Since the LEKM algorithm is an iterative algorithm, we can in practice update the cluster centers as follows:

$$z_{lj} = \frac{\sum_{i=1}^{n} u_{il} \left[1 + (x_{ij} - z_{lj}^*)^2\right]^{-1} x_{ij}}{\sum_{i=1}^{n} u_{il} \left[1 + (x_{ij} - z_{lj}^*)^2\right]^{-1}} \tag{16}$$

for $l = 1, 2, \ldots, k$ and $j = 1, 2, \ldots, d$, where $Z^* = \{\mathbf{z}_1^*, \mathbf{z}_2^*, \ldots, \mathbf{z}_k^*\}$ is the set of cluster centers from the previous iteration. When the algorithm converges, the cluster centers in the current iteration are the same as those from the previous iteration and Equation (16) is the same as Equation (15).

To find the optimal values of U, W, and Z that minimize the objective function given in Equation (10), the LEKM algorithm proceeds iteratively by updating one of U, W, and Z at a time with other other two fixed. The pseudo-code of the LEKM algorithm is shown in Algorithm 1. The computational complexity of one iteration of the LEKM algorithm is $O(nkd)$. Although the runtime complexity of the LEKM algorithm is the same as those of the EWKM algorithm and the LAC algorithm, we expect the LEKM algorithm to be slower than the EWKM algorithm and the LAC algorithm as more operations are involved in the LEKM algorithm.

The LEKM algorithm requires four parameters: k, λ, δ, and N_{max}. The parameter k is the desired number of clusters. The parameter λ controls the smoothness of the attribute weights. The larger the value of λ, the more uniform of the attribute weights. The last two parameters are used to terminate the algorithm. Table 1 gives some default values of some parameters.

4. **Numerical experiments.** In this section, we present numerical experiments based on both synthetic data and real data to demonstrate the performance of the LEKM algorithm. We also compare the LEKM algorithm with the EWKM algorithm and the LAC algorithm in terms of accuracy and runtime. We implemented all three algorithms in Java and used the same convergence criterion as shown in Algorithm 1.

Algorithm 1: Pseudo-code of the LEKM Algorithm.

Input: X, k, λ, δ, N_{max}
Output: Optimal values of U, W, and Z

Initialize $W^{(0)}$ with equal values (i.e., set $w_{lj} = 1/d$);
Initialize $Z^{(0)}$ by selecting k points from X randomly;
Update $U^{(0)}$ according to Theorem 3.1;
$s \leftarrow 0$;
$P^{(0)} \leftarrow 0$;
while *True* **do**
    Update $Z^{(s+1)}$ according to Equation (16);
    Update $U^{(s+1)}$ according to Theorem 3.1;
    Update $W^{(s+1)}$ according to Theorem 3.2;
    $s \leftarrow s + 1$;
    $P^{(s+1)} \leftarrow P\left(U^{(s+1)}, W^{(s+1)}, Z^{(s+1)}\right)$;
    if $\left|P^{(s+1)} - P^{(s)}\right| < \delta$ *or* $s \geq N_{max}$ **then**
        Break;
    end
end

Parameter	Default Value
λ	1
δ	10^{-6}
N_{max}	100

TABLE 1. Default parameter values of the LEKM algorithm.

In our experiments, we use the corrected Rand index [8, 13] to measure the accuracy of clustering results. The corrected Rand index is calculated from two partitions of the same dataset and its value ranges from -1 to 1, with 1 indicating perfect agreement between the two partitions and 0 indicating agreement by chance. In general, the higher the corrected Rand index, the better the clustering result.

Since the all the three algorithms are k-means-type algorithms, they are sensitive to initial cluster centers [6, 13]. To compare the performance of these three algorithms on the first synthetic dataset, we run these algorithm 100 times and calculate the average accuracy and runtime. In each run, we use a different seed to select random initial cluster centers. To compare the three algorithms in a consistent way, we used the same 100 seeds for all three algorithms. To test the impact of the parameters (i.e., γ in EWKM, h in LAC, and λ in LEKM), we use five different values for the parameter: 1, 2, 4, 8, and 16.

4.1. **Experiments on synthetic data.** To test the performance of the LEKM algorithm, we generated two synthetic datasets. The first synthetic dataset is a 2-dimensional dataset with two clusters and is shown in Figure 1. From the figure we see that the cluster in the top is compact but the cluster in the bottom contains

several points that are far away from the cluster center. We can consider this dataset as a dataset containing noises.

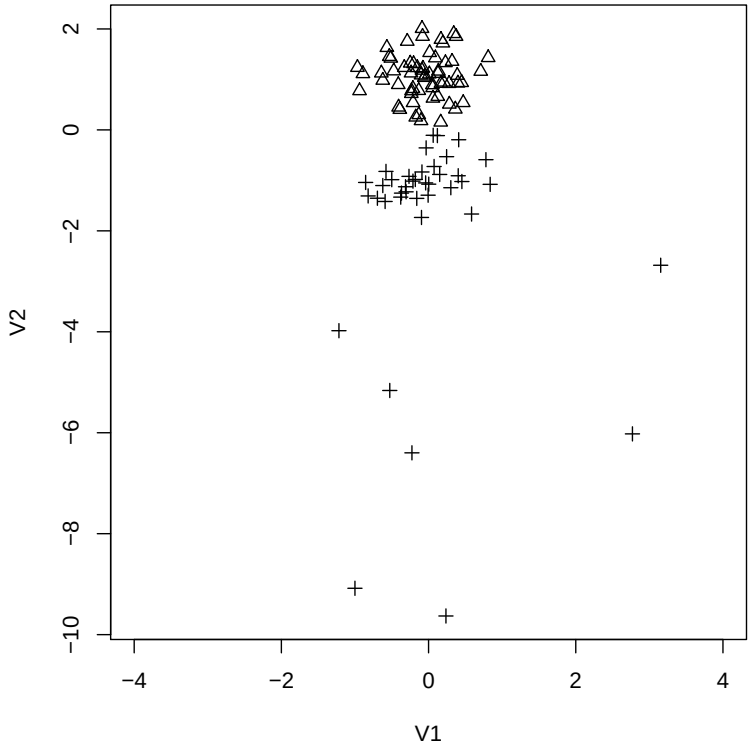

FIGURE 1. A 2-dimensional dataset with two clusters.

Parameter	EWKM	LAC	LEKM
1	0.0351 (0.0582)	0.0024 (0.0158)	0.9154 (0.2704)
2	0.0378 (0.0556)	0.9054 (0.2322)	0.9063 (0.2827)
4	0.012 (0.031)	0.8019 (0.2422)	0.9067 (0.2815)
8	-0.0135 (0.0125)	0.7604 (0.2406)	0.9072 (0.2799)
16	-0.013 (0.0134)	0.7527 (0.2501)	0.9072 (0.2799)

TABLE 2. The average accuracy of 100 runs of the three algorithms on the first synthetic dataset. The numbers in parenthesis are the corresponding standard deviations over the 100 runs. The parameter refers to γ, h, and λ in EWKM, LAC, and LEKM, respectively.

Table 2 shows the average corrected Rand index of 100 runs of the three algorithms on the first synthetic dataset. From the table we see that the LEKM algorithm produced more accurate results than the LAC algorithm and the EWKM algorithm. The EWKM produced the least accurate results. Since the dispersion of an attribute in a cluster is normalized by the size of the cluster in the LAC

	1	2
C2	35	25
C1	25	15

(a)

	1	2
C2	59	0
C1	1	40

(b)

	1	2
C2	60	0
C1	0	40

(c)

TABLE 3. The confusion matrices of the first synthetic dataset correspond to the runs with the lowest objective function values. The parameter used in these runs is 2. The labels "1" and "2" in the first row indicate the given clusters. The labels "C1" and "C2" in the first column indicate the found clusters. (a) EWKM. (b) LAC. (c) LEKM.

and LEKM algorithms, the LAC and LEKM algorithms are less sensitive to the parameter.

Table 3 shows the confusion matrices produced by the best run of the three algorithms on the first synthetic dataset. We run the EWKM algorithm, the LAC algorithm, and the LEKM algorithm 100 times on the first synthetic dataset with parameter 2 (i.e., $\gamma = 2$ in EWKM, $h = 2$ in LAC, and $\lambda = 2$ in LEKM) and chose the best run to be the run with the lowest objective function value. From Table 3 we see that the LEKM algorithm was able to recover the two clusters from the first synthetic dataset correctly. The LAC algorithm clustered one point incorrectly. The EWKM algorithm is sensitive to noises and clustered many points incorrectly.

		Weight
C1	1	3.01E-36
C2	1	2.85E-51

(a)

	Weight	
C1	0.8931	0.1069
C2	0.5057	0.4943

(b)

	Weight	
C1	0.5448	0.4552
C2	0.5055	0.4945

(c)

TABLE 4. The attribute weights of the two clusters correspond to the runs with the lowest objective function values. The parameter used in these runs is 2. The labels "C1" and "C2" in the first column indicate the found clusters. (a) EWKM. (b) LAC. (c) LEKM.

Table 4 shows the attribute weights of the two clusters produced by the best runs of the three algorithms. As we can see from the table that the attribute weights produced by the EWKM algorithm are dominated by one attribute. The attribute weights of one cluster produced by the LAC algorithm is also affected by the noises in the cluster. The attribute weights of the clusters produced by the LEKM algorithm seem reasonable as the two clusters are formed in the full space and approximate the same attribute weights are expected.

Table 5 shows the average runtime of the 100 runs of the three algorithms on the first synthetic dataset. From the table we see that the EWKM algorithm converged the fastest. The LAC algorithm and the LEKM algorithm converged in about the same time.

The second synthetic dataset is a 100-dimensional dataset with four clusters. Table 6 shows the sizes and dimensions of the four clusters. This dataset was also used to test the SAP algorithm developed in [13]. Table 7 summarizes the clustering

Parameter	EWKM	LAC	LEKM
1	0.0005 (0.0005)	0.0021 (0.0032)	0.0016 (0.0009)
2	0.0002 (0.0004)	0.0018 (0.0026)	0.0013 (0.0006)
4	0.0002 (0.0004)	0.0017 (0.0025)	0.0014 (0.0011)
8	0.0003 (0.0004)	0.0018 (0.0026)	0.0016 (0.0017)
16	0.0002 (0.0004)	0.0018 (0.0025)	0.0016 (0.002)

TABLE 5. The average runtime of the three algorithms on the first synthetic dataset. The numbers in parenthesis are the corresponding standard deviations over the 100 runs. The numbers are in seconds.

Cluster	Cluster Size	Subspace Dimensions
A	500	10,15,70
B	300	20,30,80,85
C	500	30,40,70,90,95
D	700	40,45,50,55,60,80

TABLE 6. A 100-dimensional dataset with 4 subspace clusters.

results of the three algorithms. From the table we see that the LEKM algorithm produced the most accurate results when the parameter is small. When the parameter is large, the attribute weights calculated by the LEKM algorithm become approximately the same. Since the clusters are embedded in subspaces, assigning approximately the same weight to attributes prevents the LEKM algorithm from recovering these clusters.

Parameter	EWKM	LAC	LEKM
1	0.557 (0.1851)	0.5534 (0.1857)	0.9123 (0.147)
2	0.557 (0.1851)	0.5572 (0.1883)	0.928 (0.1361)
4	0.557 (0.1851)	0.5658 (0.1902)	0.6128 (0.1626)
8	0.557 (0.1851)	0.574 (0.2028)	0.3197 (0.1247)
16	0.5573 (0.1854)	0.6631 (0.2532)	0.2293 (0.0914)

TABLE 7. The average accuracy of 100 runs of the three algorithms on the second synthetic dataset.

Table 8 shows the confusion matrices produced by the runs of the three algorithms with the lowest objective function value. From the table we see that only three points were clustered incorrectly by the LEKM algorithm. Many points were clustered incorrectly by the EWKM algorithm and the LAC algorithm. Figures 2, 3, and 4 plot the attribute weights of the four clusters corresponding to the confusion matrices given in Table 8. From Figures 2 and 3 we can see that the attribute weights were dominated by a single attribute. Figure 4 shows that the LEKM algorithm was able to recover all the subspace dimensions correctly.

Table 9 shows the average runtime of 100 runs of the three algorithms on the second synthetic dataset. From the table we see that the LEKM algorithm is slower than the other two algorithms. Since the center calculation of the LEKM algorithm

	A	B	C	D
C1	445	18	0	31
C3	17	269	0	32
C4	16	13	0	605
C2	22	0	500	32

(a)

	A	B	C	D
C2	453	8	30	29
C3	23	13	0	671
C4	24	17	470	0
C1	0	262	0	0

(b)

	A	B	C	D
C4	500	1	0	1
C1	0	299	0	0
C2	0	0	499	0
C3	0	0	1	699

(c)

TABLE 8. Confusion matrices of the second synthetic dataset produced by the runs with the lowest objective function values. In these runs, the parameter was set to 2. (a) EWKM. (b) LAC. (c) LEKM.

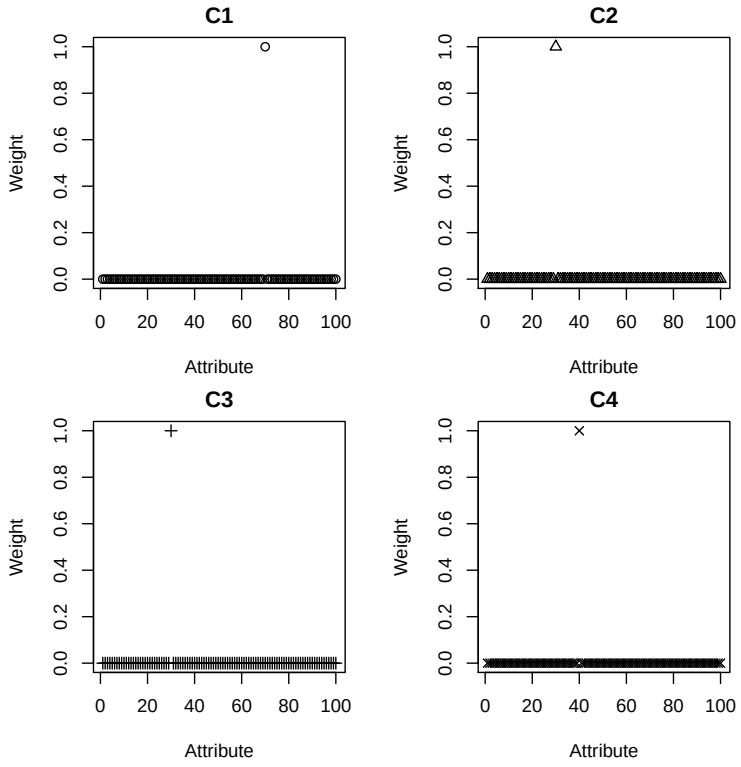

FIGURE 2. Attribute weights of the four clusters produced by the EWKM algorithm.

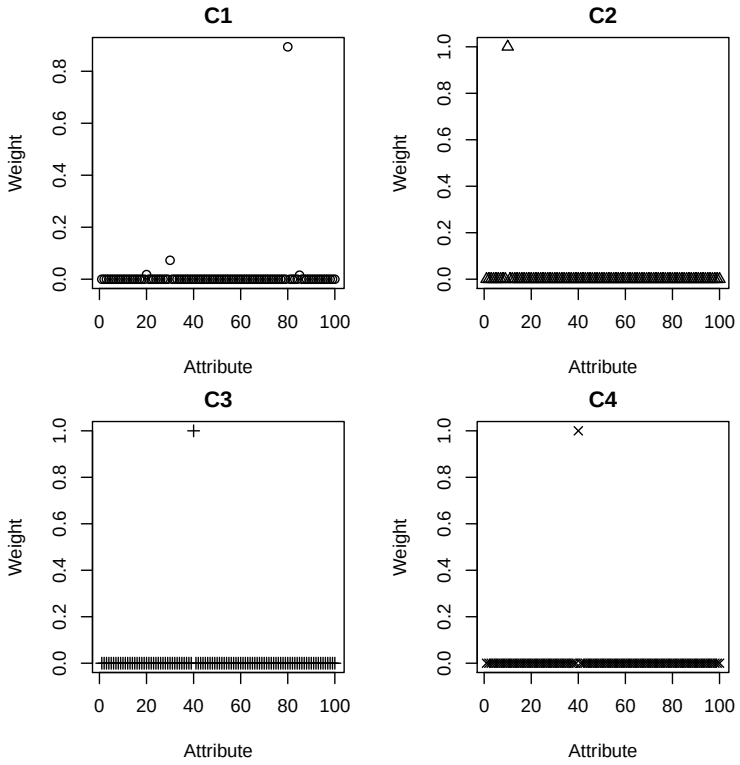

FIGURE 3. Attribute weights of the four clusters produced by the LAC algorithm.

Parameter	EWKM	LAC	LEKM
1	0.7849 (0.4221)	1.1788 (0.763)	10.4702 (0.1906)
2	0.7687 (0.4141)	0.8862 (0.4952)	10.3953 (0.1704)
4	0.7619 (0.4101)	0.8412 (0.4721)	10.5236 (0.2023)
8	0.7567 (0.4074)	0.8767 (0.4816)	10.5059 (0.2014)
16	0.7578 (0.4112)	0.8136 (0.5069)	10.4122 (0.189)

TABLE 9. The average runtime of 100 runs of the three algorithms on the second synthetic dataset.

is more complicate than that of the EWKM algorithm and the LAC algorithm, it is expected that the LEKM algorithm is slower than the other two algorithms.

In summary, the test results on synthetic datasets have shown that the LEKM algorithm is able to recover clusters from noise data and recover clusters embedded in subspaces. The test results also show that the LEKM algorithm is less sensitive to noises and parameter values that the EWKM algorithm and the LEKM algorithm. However, the LEKM algorithm is in general slower than the other two algorithm due to its complex center calculation.

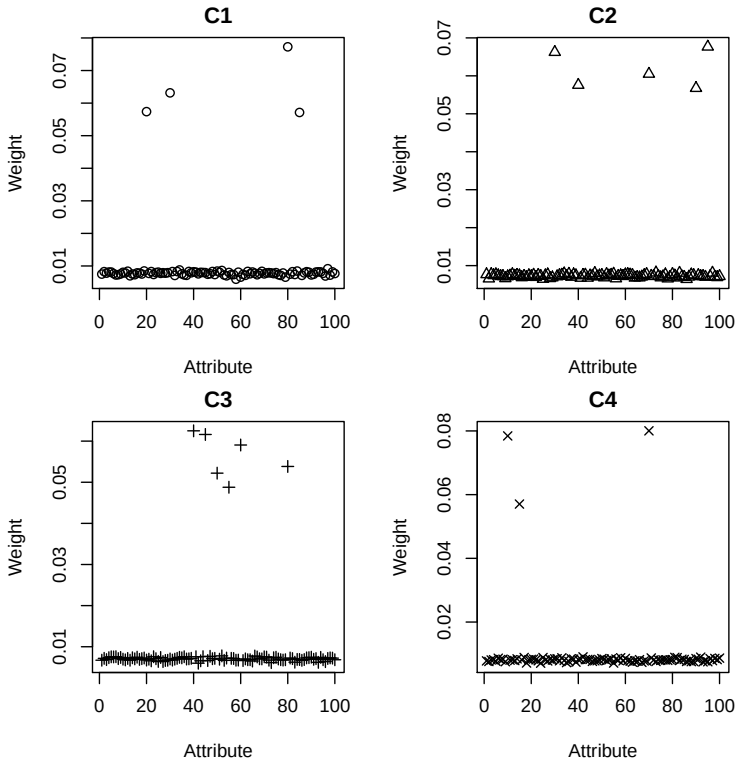

FIGURE 4. Attribute weights of the four clusters produced by the LEKM algorithm.

4.2. **Experiments on real data.** To test the algorithms on real data, we obtained two cancer gene expression datasets from [8][1]. The first dataset contains gene expression data of human liver cancers and the second dataset contains gene expression data of breast tumors and colon tumors. Table 10 shows the information of the two real datasets. The two datasets have known labels, which tell the type of sample of each data point. The two datasets were also used to test the SAP algorithm in [13].

Dataset	Samples	Dimensions	Cluster sizes
Chen-2002	179	85	104,76
Chowdary-2006	104	182	62,42

TABLE 10. Two real gene expression datasets.

Table 11 and Table 12 summarize the average accuracy and the average runtime of 100 runs of the three algorithms on the Chen-2002 dataset, respectively. From the average corrected Rand index shown in Table 11 we see that the LEKM algorithm produced more accurate results than the EWKM algorithm and the LAC algorithm did. However, the LEKM algorithm was slower than the other two algorithm.

[1]The datasets are available at `http://bioinformatics.rutgers.edu/Static/Supplements/CompCancer/datasets.htm`

Parameter	EWKM	LAC	LEKM
1	0.025 (0.0395)	0.0042 (0.0617)	0.2599 (0.2973)
2	0.0203 (0.0343)	0.0888 (0.1903)	0.2563 (0.2868)
4	0.0135 (0.0279)	0.041 (0.1454)	0.2743 (0.2972)
8	0.0141 (0.0449)	0.0484 (0.1761)	0.2856 (0.2993)
16	0.0002 (0.0416)	0.0445 (0.1726)	0.2789 (0.2984)

TABLE 11. The average accuracy of 100 runs of the three algorithms on the Chen-2002 dataset.

Parameter	EWKM	LAC	LEKM
1	0.0111 (0.0031)	0.0162 (0.0083)	0.102 (0.0297)
2	0.0123 (0.0033)	0.0124 (0.006)	0.1035 (0.0286)
4	0.0143 (0.006)	0.0151 (0.0105)	0.1046 (0.0316)
8	0.0122 (0.0043)	0.0137 (0.0089)	0.1068 (0.0337)
16	0.0144 (0.007)	0.014 (0.0091)	0.105 (0.0323)

TABLE 12. The average runtime of 100 runs of the three algorithms on the Chen-2002 dataset.

Parameter	EWKM	LAC	LEKM
1	0.3952 (0.3943)	0.5197 (0.2883)	0.5826 (0.3199)
2	0.3819 (0.3825)	0.19 (0.2568)	0.5757 (0.3261)
4	0.3839 (0.3677)	0.0772 (0.1016)	0.5823 (0.3221)
8	0.4188 (0.3584)	0.0595 (0.0224)	0.5756 (0.3383)
16	0.4994 (0.3927)	0.0625 (0.0184)	0.582 (0.3363)

TABLE 13. The average accuracy of 100 runs of the three algorithms on the Chowdary-2006 dataset.

Parameter	EWKM	LAC	LEKM
1	0.0115 (0.0048)	0.0109 (0.0042)	0.1369 (0.0756)
2	0.011 (0.0046)	0.0156 (0.0093)	0.1446 (0.0723)
4	0.0103 (0.0042)	0.0147 (0.0076)	0.1514 (0.0805)
8	0.0107 (0.005)	0.0141 (0.0063)	0.1524 (0.0769)
16	0.0113 (0.0047)	0.0138 (0.0068)	0.1542 (0.0854)

TABLE 14. The average runtime of 100 runs of the three algorithms on the Chowdary-2006 dataset.

The average accuracy and runtime of 100 runs of the three algorithms on the Chowdary-2006 dataset are shown in Table 13 and Table 14, respectively. From Table 13 we see than the LEKM algorithm again produced more accurate clustering results than the other two algorithm did. When the parameter was set to be 1, the LAC produced better results than the EWKM algorithm did. For other cases, however, the EWKM algorithm produced better results than the LAC algorithm

did. The LAC algorithm and the EWKM algorithm are much faster than the LEKM algorithm as shown in Table 14.

In summary, the test results on real datasets show that the LEKM algorithm produced more accurate clustering results on average than the EWKM algorithm and the LAC algorithm did. However, the LEKM algorithm was slower than the other two algorithms.

5. **Concluding remarks.** The EWKM algorithm [21] and the LAC algorithm [9] are two soft subspace clustering algorithms that are similar to each other. In both algorithms, the attribute weights of a cluster are calculated as exponential normalizations of the negative attribute dispersions in the cluster scaled by a parameter. Setting the parameter is a challenge when the attribute dispersions in a cluster have a large range. In this paper, we proposed the LEKM (log-transformed entropy weighting k-means) algorithm by using log-transformed distances in the objective function so that the attribute dispersions in a cluster are smaller than those in the EWKM algorithm and the LAC algorithm. The proposed LEKM algorithm has the following two properties: first, the LEKM algorithm allows users to choose a value for the parameter easily because the attribute dispersions in a cluster have a small range; second, the LEKM algorithm is less sensitive to noises because data points far away from they corresponding cluster centers are given small weights in the cluster center calculation.

We tested the performance of the LEKM algorithm and compared it with the EWKM algorithm and the LAC algorithm. The test results on both synthetic datasets and real datasets have shown that the LEKM algorithm is able to outperform the EWKM algorithm and the LAC algorithm in terms of accuracy. However, one limitation of the LEKM algorithm is that it is slower than the other two algorithm because updating the cluster centers in each iteration in the LEKM algorithm is more complicate than that in the other two algorithms.

Another limitation of the LEKM algorithm is that it is sensitive to initial cluster centers. This limitation is common to most of the k-means-type algorithms, which include the EWKM algorithm and the LAC algorithm. Other efficient cluster center initialization methods [24, 5, 6] can be used to improve the performance of the k-means-type algorithms including the LEKM algorithm.

Acknowledgments. The authors would like to thank referees for their insightful comments that greatly improve the quality of the paper.

REFERENCES

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

[2] R. Agrawal, J. Gehrke, D. Gunopulos and P. Raghavan, Automatic subspace clustering of high dimensional data for data mining applications, in *SIGMOD Record ACM Special Interest Group on Management of Data*, ACM Press, New York, NY, USA, **27** (1998), 94–105.

[3] S. Boutemedjet, D. Ziou and N. Bouguila, Model-based subspace clustering of non-gaussian data, *Neurocomputing*, **73** (2010), 1730–1739.

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

[5] F. Cao, J. Liang and G. Jiang, An initialization method for the k-means algorithm using neighborhood model, *Computers & Mathematics with Applications*, **58** (2009), 474–483.

[6] M. E. Celebi, H. A. Kingravi and P. A. Vela, A comparative study of efficient initialization methods for the k-means clustering algorithm, *Expert Systems with Applications*, **40** (2013), 200–210.
[7] X. Chen, Y. Ye, X. Xu and J. Z. Huang, A feature group weighting method for subspace clustering of high-dimensional data, *Pattern Recognition*, **45** (2012), 434–446.
[8] M. de Souto, I. Costa, D. de Araujo, T. Ludermir and A. Schliep, Clustering cancer gene expression data: A comparative study, *BMC Bioinformatics*, **9** (2008), 497–510.
[9] C. Domeniconi, D. Gunopulos, S. Ma, B. Yan, M. Al-Razgan and D. Papadopoulos, Locally adaptive metrics for clustering high dimensional data, *Data Mining and Knowledge Discovery*, **14** (2007), 63–97.
[10] B. Duran and P. Odell, *Cluster Analysis - A survey*, vol. 100 of Lecture Notes in Economics and Mathematical Systems, Springer-Verlage, Berlin, Heidelberg, New York, 1974.
[11] E. Elhamifar and R. Vidal, Sparse subspace clustering: Algorithm, theory, and applications, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35** (2013), 2765–2781.
[12] G. Gan, *Data Clustering in C++: An Object-Oriented Approach*, Data Mining and Knowledge Discovery Series, Chapman & Hall/CRC Press, Boca Raton, FL, USA, 2011.
[13] G. Gan and M. K.-P. Ng, Subspace clustering using affinity propagation, *Pattern Recognition*, **48** (2015), 1455–1464.
[14] G. Gan and M. K.-P. Ng, Subspace clustering with automatic feature grouping, *Pattern Recognition*, **48** (2015), 3703–3713.
[15] G. Gan and J. Wu, Subspace clustering for high dimensional categorical data, *ACM SIGKDD Explorations Newsletter*, **6** (2004), 87–94.
[16] G. Gan and J. Wu, A convergence theorem for the fuzzy subspace clustering (FSC) algorithm, *Pattern Recognition*, **41** (2008), 1939–1947.
[17] G. Gan, J. Wu and Z. Yang, A fuzzy subspace algorithm for clustering high dimensional data, in *Lecture Notes in Artificial Intelligence* (eds. X. Li, S. Wang and Z. Dong), vol. 4093, Springer-Verlag, 2006, 271–278.
[18] J. A. Hartigan, *Clustering Algorithms*, Wiley, New York, NY, 1975.
[19] J. Huang, M. Ng, H. Rong and Z. Li, Automated variable weighting in k-means type clustering, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27** (2005), 657–668.
[20] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.
[21] L. Jing, M. Ng and J. Huang, An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data, *IEEE Transactions on Knowledge and Data Engineering*, **19** (2007), 1026–1041.
[22] H.-P. Kriegel, P. Kröger and A. Zimek, Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering, *ACM Transactions on Knowledge Discovery from Data*, **3** (2009), 1–58.
[23] J. Macqueen, Some methods for classification and analysis of multivariate observations, in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics andProbability* (eds. L. LeCam and J. Neyman), vol. 1, University of California Press, Berkely, CA, USA, 1967, 281–297.
[24] J. Peña, J. Lozano and P. Larrañaga, An empirical comparison of four initialization methods for the k-means algorithm, *Pattern Recognition Letters*, **20** (1999), 1027–1040.
[25] L. Peng and J. Zhang, An entropy weighting mixture model for subspace clustering of high-dimensional data, *Pattern Recognition Letters*, **32** (2011), 1154–1161.
[26] R. Xu and D. Wunsch, *Clustering*, Wiley-IEEE Press, Hoboken, NJ, 2008.

Received May 2015; revised August 2015.

E-mail address: guojun.gan@uconn.edu
E-mail address: kun.chen@uconn.edu