Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 761468, 11 pages
http://dx.doi.org/10.1155/2014/761468

Research Article

Adaptive Initialization Method Based on Spatial Local Information for *k*-Means Algorithm

Honghong Liao,[1] Jinhai Xiang,[1,2] Weiping Sun,[1] Jianghua Dai,[1] and Shengsheng Yu[1]

[1] *School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*
[2] *College of Science, Huazhong Agricultural University, Wuhan 430070, China*

Correspondence should be addressed to Weiping Sun; wpsun@hust.edu.cn

Received 26 November 2013; Accepted 20 February 2014; Published 30 March 2014

Academic Editor: Yi-Kuei Lin

k-means algorithm is a widely used clustering algorithm in data mining and machine learning community. However, the initial guess of cluster centers affects the clustering result seriously, which means that improper initialization cannot lead to a desirous clustering result. How to choose suitable initial centers is an important research issue for *k*-means algorithm. In this paper, we propose an adaptive initialization framework based on spatial local information (AIF-SLI), which takes advantage of local density of data distribution. As it is difficult to estimate density correctly, we develop two approximate estimations: density by t-nearest neighborhoods (t-NN) and density by ϵ-neighborhoods (ϵ-Ball), leading to two implements of the proposed framework. Our empirical study on more than 20 datasets shows promising performance of the proposed framework and denotes that it has several advantages: (1) can find the reasonable candidates of initial centers effectively; (2) it can reduce the iterations of *k*-means' methods significantly; (3) it is robust to outliers; and (4) it is easy to implement.

1. Introduction

Clustering is a process of grouping a set of data objects into clusters based on information found in that data [1], which has a long history in a variety of scientific disciplines from statistics and computer science to biology, medicine, and even psychology. The main goals of clustering involve compressing, classifying, and gaining some useful information from data.

Clustering algorithms can be divided into two categories roughly: hierarchical and partitional [2]. Hierarchical clustering algorithms recursively find nested clusters either in agglomerative (button-up) mode or in divisive (top-down) mode, whereas partitional algorithms find clusters simultaneously as a partition of the dataset. Most hierarchical algorithms have quadratic or higher time complexity with the number of data points [3] and therefore are not suitable for large scale big data application. However, partitional algorithms often have lower time complexity and are used in many large scale tasks including bag-of-features (BoF) method in computer vision [4], color quantization in graphics and image processing [5], bag-of-words model for text classification [6], and pretraining of deep learning nowadays [7].

The *k*-means algorithm is undoubtedly one of the most popular and widely used clustering algorithms [8]. *k*-means algorithm is a hard partitional algorithm, which divides a dataset into a set of exhaustive and mutually exclusive clusters. That is, for a given dataset $\mathscr{X} = \{\mathbf{x_1}, \dots, \mathbf{x_n}\}$ in $\mathbb{R}^d$, *k*-means algorithm iteratively divides $\mathscr{X}$ into K clusters $C = \{C_1, \dots, C_K\}$, subject to $\mathscr{X} = \bigcup_{i=1}^{K} C_i$ and $C_i \cap C_j = \emptyset$, for all $1 \le i \ne j \le K$. The *k*-means algorithm usually generates clusters by optimizing a certain criterion function, and the most intuitive and frequently used one is the Sum of Squared Error (SSE) which is given by

$$\mathrm{SSE} = \sum_{i=1}^{K} \sum_{x_j \in C_i} \left\| \mathbf{x_j} - \mathbf{c_i} \right\|_2^2, \tag{1}$$

where $\| \cdot \|_2$ denotes the Euclidean norm or ℓ_2 norm and $\mathbf{c_i} = (1/|C_i|) \sum_{j \in C_i} \mathbf{x_j}$ is the center of cluster C_i whose cardinality is $|C_i|$. As a result, finding the optimal clusters for k-means algorithm turns to minimize a criterion function. Other criterion functions can also be used, such as city block (ℓ_1 norm), hamming distance, and cosine dissimilarity.

After giving the initial centers, k-means algorithm repeats two alternate procedures to group data points into K desired clusters [9]: it first properly assigns each data point to one of the K separate initial centers and then updates the clusters' center based on the assignments. The assignment and update procedures are repeated until either there is no further changes of the criterion function or the maximum number of iteration reaches.

In spite of it is popularity, k-means algorithm has some drawbacks [10]: (1) it needs the user to specify the number of clusters in advance or run independently for different values of K and then selects the partition that appears to be the most meaningful by domain experts; (2) it can only detect compact, hyperphysical clusters that are well separated and is not suitable for high dimension task because of using Euclidean distance as its default similarity metric; (3) it is sensitive to noise and outliers in datasets, for even a few of such data points can significantly influence the center (mean) of their respective cluster; (4) it often converges to a local minimum of the criterion function due to its gradient descent nature and nonconvexity of the criterion function; (5) it is highly sensitive to the selection of the initial guess of centers. Adverse effects of improper initialization include empty cluster, slower convergence, and higher chance of getting stuck in bad local minima [5]. As discussed by Celebi et al. [10], all of these drawbacks except for choosing the number of clusters can be remedied by using a suitable adaptive initialization method.

As mentioned by Celebi and Kingravi in [11], initial centers should avoid choosing outliers and being too close to each other. There are many studies focusing on initialization of k-means, such as random selection, probability-based initial methods, and factor analysis. A more intuitive idea is to determine initial centers according to the spatial distribution of data points. That is, if we choose initial centers from regions with high local density of data distribution, which is a kind of spatial local information of data points, outliers will be prevented from being chosen. And, by keeping them away with certain distance, we will get the suitable initial centers for k-means algorithm (as shown in Figure 2).

Inspired by the discussion above, we propose an adaptive initialization framework based on spatial local information (AIF-SLI), which takes advantage of local density of data distribution, and develop two approximate estimations for density, that is, density by t-nearest neighborhoods (t-NN) and density by ϵ-neighborhoods (ϵ-Ball), leading to two implements of the proposed framework. Both implements need three steps. Firstly, we have to find out the high density regions of data points based on the estimation of local data density. Secondly, we need to mark data points that belong to high density regions as candidates of initial centers. And thirdly, we should determine which candidates should be selected as initial centers and make sure that they are kept away with certain distance.

The contributions of this paper are as follows: (1) we propose an adaptive framework of initial guess of clusters' center based on spatial local information; (2) we derive two implements of the proposed framework; (3) we give a comparative empirical study among the proposed framework and the state-of-the-art techniques and analyse the rationality of the proposed framework.

Our empirical study on more than 20 datasets shows promising performance of the proposed framework and denotes that it has several advantages: (1) it can find the suitable candidates of initial centers effectively; (2) it can reduce the iteration of k-means methods significantly; (3) it is robust to outliers; and (4) it is easy to implement.

The rest of this paper is organized as follows. Section 2 reviews the related work in the literature of initialization for k-means algorithm. Section 3 gives the proposed adaptive initialization framework based on spatial local information. The two implements of the proposed framework, t-nearest neighborhoods (t-NN) and ϵ-neighborhoods (ϵ-Ball), are derived in Section 4. Section 5 describes the datasets, feature normalization, the performance criteria, and parameter settings in our experiments. Section 6 denotes our extensive empirical study of evaluating the performance of the proposed method. Section 7 analyses the complexity for our method and the suitability of initialing k-means algorithm based on spatial local information. After that, we conclude this work.

2. Related Work

There is a considerable amount of literature on the initialization methods of k-means algorithm; we briefly review some of the commonly used ones. A comprehensive review can be found in recent work by Celebi et al. [10].

There are a number of initialization methods selecting the initial centers in a probability manner, implicitly or explicitly. The most used initial method for k-means algorithm is MacQueen's second initialization, that is, selecting the initial centers randomly [12]. Due to its random nature, it inevitably selects outliers as the initial centers and leads to more iteration or bad local minima. k-means based on this initial method usually needs to run several times with multiple different initial partitions and chooses the partition with the smallest squared error. Forgy's method [13] is a random assignment method, which first assigns each point to one of the K clusters uniformly. Then the initial centers are obtained by centroids of data points according to their assignment. This method usually confuses with MacQueen's second method discussed above in which the first step is selecting initial centers not doing assignment. Both methods can be viewed as selecting (assigning) under the uniform distribution of data points. The k-means++ method [14] chooses the first center arbitrarily and the ith center ($i \in \{2, 3, \ldots, K\}$) with a probability of $\mathrm{md}(\mathbf{x}')^2/(\sum_{i=1}^{n} \mathrm{md}(\mathbf{x}_i)^2)$, where $\mathrm{md}(\mathbf{x})$ denotes the minimum distance from a point $\mathbf{x}$ to the previously selected centers. It implies that data points with larger distance to the selected centers have higher probability to be chosen as new initial centers. A parallel version of k-means++ algorithm was developed in [15].

Several works consider choosing initial centers with a desired distance apart from each other. Ball and Hall's method [16] takes the centroid of dataset $\mathscr{X}$ as the first initial center; that is, $c_1^0 = (1/|\mathscr{X}|) \sum_{\mathbf{x}_i \in \mathscr{X}} \mathbf{x}_i$. It then takes the data point, which is at least T units apart from the previously selected centers, as initial center until K centers are obtained. This method chooses data points with a desired distance apart from each other and at the same time prevents initial centers from becoming too close to each other. However, it is difficult to determine a reasonable threshold T. Maximin method [17, 18] takes the first center arbitrarily in dataset $\mathscr{X}$. And the second initial center is chosen with the largest distance from the first center. Other initial centers are chosen to be the points with the greatest minimum distance to the previously selected centers. It should be noted that in the work of [18], it chooses the data point with maximum norm in dataset $\mathscr{X}$ as the first center. Erisoglu et al. [19] choose two of d features that describe the changes best in the dataset as main axes, where d is the dimension of the space where dataset $\mathscr{X}$ lies. All initial centers are chosen based on the main axes by projecting dataset $\mathscr{X}$ to main axes. The data point with maximum distance from the mean of dataset is chosen as the first initial center. The ith center is chosen with the largest sum of distance from previously selected centers, until K centers are obtained. They cannot guarantee not selecting outliers as initial centers.

Recently, researches utilize factor analysis method to initialize k-means algorithm. The PCA-Part method [20] uses a divisive hierarchical approach based on Principal Component Analysis (PCA). This method iteratively selects the cluster with the greatest SSE and divides it into two subclusters by a hyperplane that passes through the cluster center and orthogonal to the direction of the principal eigenvector of related covariance matrix. In the first step, it takes the entire dataset as the cluster with the greatest SSE. The Var-Part method [20] is an approximation to PCA-Part, which assumes that the covariance matrix is diagonal. In this case, the direction of hyperplane is orthogonal to the coordinate axis with the greatest variance. Celebi and Kingravi [11] use Otsu's algorithm to get an adaptive threshold for PCA-Part and Var-Part algorithm [20], in which original takes the cluster center as threshold to divide a cluster into two subclusters. This leads a deterministic initialization method for k-means and experiments to show that it gets promising performance compared to the original algorithm. Onoda et al. [21] acquire initial centers by independent components analysis (ICA). They first calculate the K independent components of $\mathscr{X}$ and then choose the point that has the least cosine distance from the ith independent components as the ith ($i \in \{1, 2, \ldots, K\}$) center.

Al-Daoud's method [22] first uniformly partitions the data space into M grids. It takes K_m points as initial centers from grid m, $m = 1, \ldots, M$, where $K_m = KN_m/N$, N_m is the number of data points in grid m and N is the total number of data points in dataset $\mathscr{X}$. This method can be viewed as a density-based initialization method. However, this method would ignore clusters with fewer number of data points and it is not easy to decide how many grids M are suitable.

The proposed initialization method bears some similarity to that of Al-Daoud and Robetrs [22], in which local density of data points is taken into consideration when determining initial centers. However, the proposed method differs remarkably from Al-Daoud's method. In their work, local density is estimated based on artificial and hard grid segmentation; different grid segmentation leads to different local density estimation. Our method estimates local density of data points according to their local spatial distribution, which can be regarded as a nature and soft group segmentation, and initial centers are determined by the distribution of data points.

3. Proposed Framework of AIF-SLI

Since reasonable initial centers should avoid choosing outliers and being too close to each other, we propose an adaptive initialization framework based on spatial local information (AIF-SLI) for k-means algorithm.

AIF-SLI takes advantage of local density of data points; that is, for a given dataset $\mathscr{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ in $\mathbb{R}^d$, we first estimate its local density via defining a function $d(\mathbf{x})$ to describe the local density for each data point $\mathbf{x} \in \mathscr{X}$. For point $\mathbf{x}$, $d(\mathbf{x})$ describes the compact of data points within a small region containing $\mathbf{x}$ as its inner point. After that, we then find out regions with density higher than a threshold ω. Initial centers of k-means algorithm are chosen from these regions with higher local density.

Generally, the proposed initialization framework involves three steps: (1) estimating local density for each data point $x \in \mathscr{X}$; (2) finding out regions with density higher than a threshold and marking data points in these regions as candidates for initial centers; (3) determining initial centers from candidates and making sure that the initial centers are separate with certain distance. The workflow of the proposed framework is demonstrated in Figure 1.

It should be noticed that although being used to initialize k-means algorithm, the proposed framework can be easily extended to other clustering algorithms such as the Gaussian Mixture Model.

4. Two Implements for AIF-SLI

In this section, we denote the two implements of the proposed framework: based on t-nearest neighborhoods (t-NN) and based on ϵ-neighborhoods (ϵ-Ball), leading to two adaptive initialization methods. We also give an algorithm to determine initial centers from candidates.

4.1. AIF-SLI Based on t-Nearest Neighborhoods (t-NN). In this subsection, we denote an approximate estimation for local density of dataset by t-nearest neighborhoods (t-NN). It is well known that t-NN is a natural choice for the approximate estimation of local density. Inspired by the superiority of Laplacian eigenmaps [23], we first construct an adjacency graph by t-NN and obtain a Gram matrix from

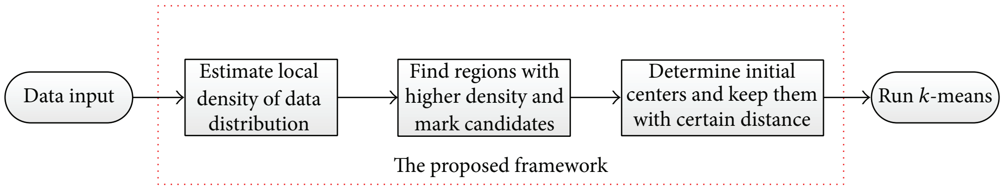
Data input
Estimate local density of data distribution
Find regions with higher density and mark candidates
Determine initial centers and keep them with certain distance
Run k-means
The proposed framework

Figure 1: The workflow of the proposed framework.

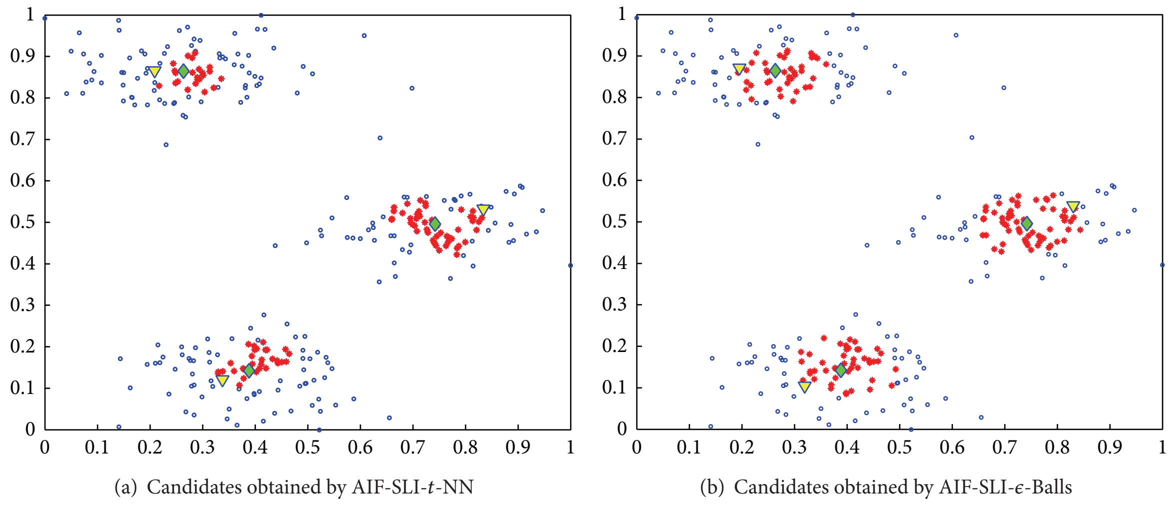

Figure 2: Rationality analysis of the proposed two algorithms derived from the framework of AIF-SLI. The red points denote the candidates, the yellow triangles are the initial centers chosen from candidates, and the final centers (centers after clustering) are presented with green diamonds.

the constructed graph. Gram matrix can be precomputed and loaded to memory before clustering. Values for Gram matrix is computed by

$$W_{ij} = \begin{cases} \exp\left\{-\dfrac{\left\|x_i - x_j\right\|^2}{\left\|x_i\right\| \cdot \left\|x_j\right\|}\right\} & \text{if } x_j \in N_t\left(x_i\right) \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where $N_t(x_i)$ denotes the t-nearest neighborhood of x_i.

Then, we introduce a vector $\vec{d}$ with n components whose entries are given by $d_i = \sum_{j=1}^{n} W_{ij}$. The vector $\vec{d}$ provides a natural measure of local density of data points: for data point x_i, if the value of d_i is larger, the local density is higher.

Algorithm 1 shows the detailed steps of approximate version of the AIF-SLI method based on t-nearest neighborhoods (t-NN). In step (4), the median($\vec{d}$) denotes the median value of vector $\vec{d}$ (in our experiments, we use the median function in MATLAB); other criteria can also be used to determine the threshold ω, such as the mean or certain confidence interval of quantile.

4.2. AIF-SLI Based on ϵ-Neighborhoods (ϵ-Ball). In this subsection, we denote an approximate estimation for local density of dataset by ϵ-neighborhoods (ϵ-Ball). We also construct an adjacency graph and Gram matrix. Values for Gram matrix are computed by

$$W_{ij} = \begin{cases} 1 & \text{if } \left\|x_i - x_j\right\| < \epsilon \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

The definition above of (3) seems much more intuitive than the definition of (2). Equation (3) counts the number of data points in a small region which contains $\mathbf{x}_i$ as its center. More numbers in a region seem more compact of data points in that region. Here, the value of ϵ is also an issue that needs to be considered, and we set it to a weighted half of the average distance of points in $\mathcal{X}$; that is, $\epsilon = 0.5 \times$ Mean Dist. The Mean Dist means average distance of points in $\mathcal{X}$ which is computed as follow:

$$\text{Mean Dist} = \frac{\sum_{i,j=1}^{n} d\left(x_i, x_j\right)}{n \times (n-1)}, \quad (4)$$

where $d(x_i, x_j)$ is the distance of data points x_i and x_j.

Algorithm for density by ϵ-neighborhoods (ϵ-Ball), which we give its name as AIF-SLI-ϵ-Ball, describes the same steps as Algorithm 1, and its details are shown in Algorithm 2.

```
Input: The dataset $\mathcal{X}$.
Output: Candidate set $C_{\text{cand}}$.
(1) Initial candidate set $C_{\text{cand}} = \emptyset$;
(2) Construct the Gram matrix $W_{ij}$ based on (2);
(3) Compute vector $\vec{d}$ with $d_i = \sum_{j=1}^{n} W_{ij}$;
(4) Determine a threshold $\omega = \text{median}(\vec{d})$;
(5) for all data points in $\mathcal{X}$
(6)     if data point $x_j$ with local density $d_j > \omega$,
(7)         Mark data point $x_j$ as a candidate initial center, that is $C_{\text{cand}} = C_{\text{cand}} \cup x_j$;
(8)     end if
(9) end for
```

Algorithm 1: AIF-SLI-t-NN.

```
Input: The dataset $\mathcal{X}$.
Output: Candidate set $C_{\text{cand}}$.
(1) Initial candidate set $C_{\text{cand}} = \emptyset$;
(2) Construct the Gram matrix $W_{ij}$ based on (3);
(3) Compute vector $\vec{d}$ with $d_i = \sum_{j=1}^{n} W_{ij}$;
(4) Determine a threshold $\omega = \text{median}(\vec{d})$;
(5) for all data points in $\mathcal{X}$
(6)     if data point $x_j$ with local density $d_j > \omega$,
(7)         Mark data point $x_j$ as a candidate initial center, that is $C_{\text{cand}} = C_{\text{cand}} \cup x_j$;
(8)     end if
(9) end for
```

Algorithm 2: AIF-SLI-ϵ-Balls.

```
Input: cluster number $K$, candidate set $C_{\text{cand}}$ (obtained by Algorithm 1 or Algorithm 2).
Output: Initial centers $\{\mathbf{c}_1^0, \dots, \mathbf{c}_K^0\}$.
(1) $\mathbf{c}_1^0 = x_i$, $x_i$ is random chosen from $C_{\text{cand}}$, set $k = 1$;
(2) $\mathbf{c}_2^0 = \arg\max_{x_j} \left\| x_j - \mathbf{c}_1^0 \right\|$, $\forall x_j \in C_{\text{cand}}$, set $k = 2$;
(3) if $k < K$ do
(4)     set $k = k + 1$, $\mathbf{c}_k^0 = \arg\max_{x_j} \left\{ \min \left\| x_j - \mathbf{c}_t^0 \right\|, \forall x_j \in C_{\text{cand}}, t = 1, \dots, k-1 \right\}$;
(5) repeat step (4) until $k = K$.
```

Algorithm 3: Determining initial centers from candidates.

4.3. Determining Initial Centers from Candidates. In this subsection, we describe the details of how to determine initial centers from candidates. We propose an algorithm very similar to the Maximin method [17] in the candidates set obtained by Algorithm 1 or Algorithm 2. Algorithm 3 shows the detailed steps of the proposed algorithm. Without loss of generality, we choose the first data point in candidates set as the first initial center in our experiments, which makes our algorithm deterministic. It should be noted that any other methods such as k-means++ can also be used to determine initial centers from candidates in our work and lead to promising performance.

5. Experiment Setup

5.1. Dataset Description. We use 25 common datasets from the UCI machine learning repository [24] to perform our experiments. Table 1 gives the descriptions for these UCI datasets. For each dataset, the number of clusters (K) is set to be equal to the true number of classes (K'), as commonly seen in the related literature [9–11, 20].

5.2. Performance Criteria. As in [10, 11], the performance of the initialization methods is quantified using two effectiveness (quality) and one efficiency (speed) criteria.

Table 1: UCI dataset descriptions (N: number of data points, D: number of data dimensions, and K': number of classes).

ID	Dataset	N	D	K'
1	Abalone	4177	7	29
2	Breast cancer Wisconsin (diagnostic)	699	9	2
3	Breast tissue	106	9	6
4	*E. coli*	336	7	8
5	Glass identification	214	9	6
6	Haberman's survival	306	3	2
7	ILPD (Indian liver patient dataset)	583	10	2
8	Image segmentation	2310	19	7
9	Ionosphere	351	34	2
10	*Iris*	150	4	4
11	Heart disease (processed Cleveland)	303	13	5
12	ISOLET 5	1559	617	26
13	Letter recognition	20000	16	26
14	Libras movement	360	90	15
15	Lung cancer	32	56	3
16	Multiple features (Fourier)	2000	76	10
17	Musk (version 1, clean 2)	6598	166	2
18	One hundred plant species leaves dataset	1600	64	100
19	Page blocks classification	5473	10	5
20	Parkinson's disease	195	22	2
21	Pima Indians diabetes	768	8	2
22	Semeion handwritten digit	1593	256	10
23	Spambase	4601	57	2
24	Wine	178	13	3
25	Yeast	1484	8	10

(i) *Initial SSE.* This is the SSE value calculated after the initialization phase, before the clustering phase. It gives us a measure of the effectiveness of an initialization method by itself.

(ii) *Final SSE.* This is the SSE value calculated after the clustering phase. It gives us a measure of the effectiveness of an initialization method when its output is refined by k-means; lower value infers compact clusters. Note that SSE value is the objective function of k-means algorithm, that is, (1).

(iii) *Number of Iterations.* This is the number of iterations that k-means requires until reaching convergence when initialized by a particular initialization method. It is an efficiency measure independent of programming language, implementation style, compiler, and CPU architecture.

All methods compared in this paper are implemented using MATLAB and executed on a PC with 2.5 GHz CPU and 4 GB RAM.

5.3. Attributes Normalization. Feature normalization is a common preprocessing step in computer vision and machine learning community. Normalization is necessary to prevent features with large ranges from dominating the distance calculations and also to avoid numerical instabilities in the computations. Two commonly used normalization schemes are linear scaling to unit range (min-max normalization) and linear scaling to unit variance (Z-score normalization). Min-max normalization is shown as follows:

$$\hat{x} = \frac{(x - x_{\min})}{(x_{\max} - x_{\min})}, \quad (5)$$

where $x_{\min}, x_{\max}$ denote the minimum and maximum values of the feature, respectively. Min-max normalization scales the attribute to the $[0, 1]$ interval.

Z-score normalization approximately maps the attribute almost to the $[-2, 2]$ interval using

$$\hat{x} = \frac{(x - \mu)}{\sigma}, \quad (6)$$

where μ and σ denote the mean and variance of the feature, respectively.

Several studies reveal that the former is more preferable than the latter in clustering research since the latter is likely to eliminate valuable between-cluster variation [20, 25]. In this paper, we apply the min-max normalization to map the attributes of each dataset to the $[0, 1]$ interval.

Table 2: Final SSE by different choices of t in AIF-SLI-t-NN.

ID	$0.1 \times n/K$	$t = 1$	5	10	20	50	100
1	14	33.93	34.03	33.91	**33.89**	33.96	34.41
4	4	18.43	18.51	18.03	**18.07**	26.38	26.92
8	33	496.4	419.9	419.9	415.7	**405.0**	409.1
13	77	2759	2764	2811	2792	**2752**	2762
19	109	227.4	227.4	215.8	215.8	215.8	**215.5**
25	15	68.87	68.60	68.17	60.99	**60.27**	68.48

Table 3: Initial SSE comparison of the initialization methods. The optimal results are marked in bold font and the second optimal results in italic font. The last line summarizes the number of optimal and the second optimal results.

ID	Random	Kpp	PCA-Part	Var-Part	Maximin	AIF-SLI-t-NN	AIF-SLI-ϵ-Balls
1	98.95	*64.81*	633.92	736.45	145.47	**62.17**	70.40
2	700.7	495.9	**239.3**	*247.5*	610.2	312.8	364.4
3	20.61	*14.05*	20.23	14.39	17.73	16.41	**13.65**
4	39.73	36.49	42.12	**25.31**	76.49	36.04	*35.38*
5	42.25	34.33	*31.73*	**28.36**	116.55	33.73	34.23
6	47.52	43.45	*26.32*	**25.28**	170.04	40.02	44.26
7	187.38	234.37	*104.54*	**90.96**	742.78	150.22	111.38
8	924.8	**761.2**	958.7	959.4	1616.8	782.7	*771.4*
9	1224.3	1357.7	*638.0*	**631.7**	1791.1	1040.5	1035.6
10	20.01	18.16	**9.57**	*10.31*	20.23	11.33	12.06
11	469.95	469.84	*339.96*	**300.45**	566.83	449.29	454.22
12	46083.9	46086.6	*41464.3*	**40860.2**	54989.6	42273.2	42909.1
13	4542.9	4522.2	5700.7	5803.9	7583.4	**2234.9**	*2262.1*
14	*909.10*	**908.60**	1253.6	992.17	1113.8	941.02	917.97
15	301.92	294.30	*185.31*	**165.57**	320.80	290.01	297.83
16	5500.3	5600.1	*4033.4*	**3781.0**	7820.8	5247.2	5150.6
17	80298.8	75099.1	*45909.6*	**37333.9**	107164.8	81283.7	81203.6
18	958.25	**948.95**	1789.2	1343.8	1023.4	960.67	*949.54*
19	442.30	398.15	**327.76**	467.08	2343.1	*352.09*	392.41
20	176.32	177.84	**99.45**	*121.73*	355.85	160.15	169.54
21	238.21	229.15	*129.17*	**124.27**	666.08	218.43	210.69
22	123059	124883	*77513*	**76608**	128445	122645	121506
23	1088.1	1171.6	*783.89*	**782.23**	13155	1035.1	1023.6
24	109.84	108.12	*60.37*	**52.24**	185.47	93.07	93.77
25	107.66	110.98	127.61	125.42	260.78	**101.84**	*107.22*
	0/*1*	**3**/*2*	**4**/*13*	**14**/*3*	**0**/*0*	**3**/*1*	**1**/*5*

5.4. Parameter Settings. In our AIF-SLI-t-NN algorithm, how to choose the number of neighborhood, t, is an important issue. We analyse the final SSE by different choices of the value of t on some datasets described in Table 1, and the results are demonstrated in Table 2.

From this table, we set $t = \max\{20, 0.1 \times n/K\}$ in our experiments, which shows neither less discriminatory nor sensitive to outliers. And the convergence of k-means was controlled by two criteria: the number of iterations reaches a maximum allowable value (we set the maximum value to be 100 in our experiments) or the relative improvement in SSE between two consecutive iterations drops below a threshold; that is, $|\text{SSE}_{i+1} - \text{SSE}_i|/\text{SSE}_i < \epsilon$, and $\epsilon = 10^{-6}$.

6. Experimental Results

In this section, we compare two implements of our AIF-SLI method: AIF-SLI-t-NN and AIF-SLI-ϵ-Balls, with five state-of-the-art methods: MacQueen's second method (random) [12], Maximin method (Maximin) [18], k-means++ (Kpp) [14], Var-Part [20], and PCA-Part [20]. All seven methods are executed 100 times and the *mean* is collected for each performance criterion. Tables 3, 4, 5, and 6 denote the performance measurements for each method with respect to initial SSE, final SSE, percentage comparison for final SSE, and number of iterations, respectively. We should note that Maximin method (Maximin) [18], Var-Part [20], PCA-Part

Table 4: Final SSE comparison of the initialization methods.

ID	Random	Kpp	PCA-Part	Var-Part	Maximin	AIF-SLI-t-NN	AIF-SLI-ϵ-Balls
1	37.10	*34.04*	235.01	591.65	38.45	**33.89**	34.63
2	237.99	237.99	237.99	237.99	237.99	237.99	237.99
3	8.41	7.73	17.28	13.73	**6.85**	7.40	*7.34*
4	*18.24*	18.39	32.67	21.94	20.04	**18.07**	18.47
5	20.07	**19.57**	29.15	25.88	23.26	19.87	*19.60*
6	25.32	*25.31*	25.37	**25.28**	25.38	25.33	25.34
7	90.96	90.96	90.96	90.96	90.96	90.96	90.96
8	*416.97*	417.98	473.34	745.33	443.42	**414.70**	417.87
9	628.90	628.90	628.89	628.90	628.90	628.89	628.90
10	6.75	**5.65**	7.13	7.13	6.06	5.91	*5.87*
11	258.41	256.95	310.94	300.21	254.87	*241.04*	**236.21**
12	24660.8	*24626.6*	35349.4	31272.5	24880.1	**24565.3**	24648.7
13	2764.0	**2742.0**	4519.6	5358.1	2783.4	*2744.9*	2747.2
14	519.57	*508.70*	1111.4	820.47	**502.43**	512.43	511.13
15	161.67	162.11	185.31	164.10	161.87	**160.09**	*161.20*
16	3175.2	3183.2	3635.2	3252.8	3283.7	**3169.5**	*3174.8*
17	*36944.3*	*36944.3*	38278.1	**36372.7**	38278.1	*36944.3*	37325.4
18	516.53	515.14	1378.4	1024.5	**508.61**	*511.14*	514.10
19	216.66	*216.42*	242.76	443.21	294.96	**215.50**	217.59
20	98.67	98.67	98.67	98.67	98.67	98.67	98.67
21	121.25	122.88	121.25	121.25	121.25	121.25	123.60
22	66656	66691	67472	67461	**66387**	66797	*66655*
23	775.6	784.3	*777.5*	*777.5*	**764.7**	*777.5*	780.73
24	48.97	48.97	48.98	48.96	48.96	48.97	48.96
25	68.10	*64.12*	113.16	113.15	69.35	**60.99**	66.13
	0/*3*	**3**/*7*	**0**/*1*	**2**/*1*	**5**/*0*	**8**/*5*	**1**/*6*

[20], and our two algorithms AIF-SLI-t-NN and AIF-SLI-ϵ-Balls are deterministic algorithms.

Table 3 shows that PCA-Part and Var-Part obtaining lowest initial SSE over other methods at most datasets, and our two algorithms achieve second optimal value. This may be because PCA-Part and Var-Part take the mean of each feature as threshold. Table 4 shows PCA-Part and Var-Part obtaining largest final SSE over other methods, which infers that PCA-Part and Var-Part cannot get compact clusters after clustering. Table 4 also shows that AIF-SLI-t-NN and AIF-SLI-ϵ-Balls get lowest final SSEs on most datasets, which infers that they obtain more compact clusters after clustering than other methods. Table 5 describes the percentage comparison for final SSE. We take the Kpp as the baseline algorithm, which is demonstrated by number "1"; the values in this table denote the percentage of algorithms compared to Kpp; the symbol "+" ("−") denotes that its value is larger (smaller) than Kpp. Take the dataset 4 as an example; the value −0.8% under algorithm *random* denotes that the final SSE by random is smaller than that of Kpp by 0.8%, and +9.0% under algorithm *Maximin* denotes that the final SSE by Maximin is larger than that of Kpp by 9.0%. The last line of this table denotes the average percentage of these algorithms compared to Kpp, and this line shows that both AIF-SLI-t-NN and AIF-SLI-ϵ-Balls get better final SSE than Kpp. Table 6 describes the number of iterations from after initialization until k-means algorithm terminates. This table implies that AIF-SLI-t-NN and AIF-SLI-ϵ-Balls converge quickly at most datasets.

From the experiment results, the proposed two algorithms reduce the number of iterations for k-means methods significantly, and from final SSE results in Tables 4 and 5, both proposed algorithms get compact clusters, which infers that our initial centers are suitable for k-means algorithm, and this suitability or reasonableness comes from the advantage of spatial local information. The proposed two algorithms are approximate estimation of spatial local information; if we make use of more elegant approaches to estimate the spatial local information, better results may be obtained.

7. Discussion

7.1. Complexity Analysis. Table 7 gives the average CPU time over 100 times executed by seven algorithms mentioned above under a subset of datasets in Table 1 (as the CPU time is dependent on implementation style, compiler, and CPU architecture, we do not add it as a performance criterion in our experiments). The CPU time is taken by preprocessing (including the calculation of Gram matrix), the initialization and the k-means running time.

Table 5: Percentage comparison for final SSE.

ID	Random	Kpp	PCA-Part	Var-Part	Maximin	AIF-SLI-t-NN	AIF-SLI-ϵ-Balls
1	+9.0%	*1*	+590%	+1638%	+12.9%	**−0.4%**	+1.7%
2	+0.0%	1	+0.0%	+0.0%	+0.0%	+0.0%	+0.0%
3	+8.8%	1	+123.5%	+77.6%	**−11.4%**	−4.3%	*−5.1%*
4	*−0.8%*	1	+77.7%	+19.3%	+9.0%	**−1.8%**	+0.4%
5	+2.6%	**1**	+49.0%	+32.2%	+18.9%	+1.5%	*+0.2%*
6	+0.0%	*1*	+0.2%	**−0.1%**	+0.3%	+0.1%	+0.1%
7	+0.0%	1	+0.0%	+0.0%	+0.0%	+0.0%	+0.0%
8	*−0.2%*	1	+13.2%	+78.3%	+6.1%	**−0.8%**	−0.0%
9	+0.0%	1	−0.0%	+0.0%	+0.0%	−0.0%	+0.0%
10	+19.5%	**1**	+26.2%	+26.2%	+7.3%	+4.6%	*+3.9%*
11	+0.6%	1	+21.0%	+16.8%	−0.8%	*−6.2%*	**−8.1%**
12	+0.1%	*1*	+43.5%	+27.0%	+1.0%	**−0.3%**	+0.1%
13	+0.8%	**1**	+64.8%	+95.4%	+1.5%	*+0.1%*	+0.2%
14	+2.1%	*1*	+118%	+61.3%	**−1.2%**	+0.7%	+0.5%
15	−0.3%	1	+14.3%	+1.2%	−0.2%	**−1.3%**	*−0.6%*
16	−0.3%	1	+14.2%	+2.2%	+3.2%	**−0.4%**	*−0.3%*
17	*0.0%*	*1*	+3.6%	**−1.6%**	+3.6%	*0.0%*	+1.0%
18	+0.3%	1	+168%	+98.9%	**−1.3%**	*−0.8%*	−0.2%
19	+0.1%	*1*	+12.2%	+105%	+36.3%	**−0.4%**	+0.5%
20	+0.0%	1	+0.0%	+0.0%	+0.0%	+0.0%	+0.0%
21	−1.3%	1	−1.3%	−1.3%	−1.3%	−1.3%	+0.6%
22	−0.1%	1	+1.2%	+1.2%	**−0.5%**	+0.2%	*−0.1%*
23	−1.1%	1	*−0.9%*	*−0.9%*	**−2.5%**	*−0.9%*	−0.5%
24	+0.0%	1	+0.0%	−0.0%	−0.0%	+0.0%	−0.0%
25	+6.2%	*1*	+76.5%	+76.5%	+8.2%	**−4.9%**	+3.1%
Average	+1.84%	1	+56.6%	+94.1%	+3.6%	−0.7%	−0.1%

In Table 7, for datasets with small scale datasets and low dimension, as datasets 4 and 5, all algorithms run quickly. However, when the number of data points is large but dimension is low, such as datasets 1, 13, and 25, both algorithms, AIF-SLI-t-NN and AIF-SLI-ϵ-Balls, need more time to execute than others. As the dimension is larger, taking datasets 8, 18, and 22 as an example, PCA-Part algorithm takes the longest running time, followed by algorithms AIF-SLI-t-NN, Var-Part, and AIF-SLI-ϵ-Balls. The AIF-SLI-ϵ-Balls algorithm always runs quicker than AIF-SLI-t-NN, which is due to the lower computational complexity of finding the ϵ-Balls than that of t-nearest neighborhood.

Table 8 shows the CPU time of a single running for each component in AIF-SLI-t-NN algorithm; these components include calculating the vector $\vec{d}$ (including constructing the Gram matrix), initialization, and running k-means. Calculating the vector $\vec{d}$ takes more than 90% of the running time for large datasets, because of finding the t-nearest neighborhoods for each data point, whose computational complexity is $O(n^2)$, time consuming.

It is should be noted that the Gram matrix has dimension n^2, which can be very large for many applications when n is of the order of a million or impossible for cases when n is of the order of a billion. In our experiments, we use the single command in MATLAB to change the data points from 8 bytes to 4 bytes to save the memory, especially for dataset 13. And also, if the parameter t in t-nearest neighborhood algorithm is small enough, the Gram matrix can be very sparse and a sparse matrix can be constructed to replace the full Gram matrix. Computational complexity and optimization of memory of the proposed method are worthy of study when the Gram matrix is sparse.

The AIF-SLI-ϵ-Balls algorithm has similar characteristic to AIF-SLI-t-NN, and we omit its analysis for the limited space.

7.2. Rationality Analysis. In this section, we denote the rationality analysis of the proposed framework. Figure 2 shows a dataset generated by three Gaussian functions with different means and variance. Figure 2(a) denotes the candidates obtained by AIF-SLI-t-NN (Algorithm 1) and Figure 2(b) denotes candidates obtained by AIF-SLI-ϵ-Balls (Algorithm 2).

From Figure 2, we confirm that, by utilizing the spatial local information, which takes advantage of local density of dataset, the initial centers can be chosen from regions with high local density and avoid choosing outliers as initial centers, which is reasonable as discussed above. At the same time, the initial centers are very close to the final centers

Table 6: Number of iterations comparison of the initialization methods.

ID	Random	Kpp	PCA-Part	Var-Part	Maximin	AIF-SLI-t-NN	AIF-SLI-ϵ-Balls
1	50.2	49.0	49.0	**10.0**	54.0	*34.0*	35.0
2	4.4	4.9	*3.0*	*3.0*	6.0	*3.0*	**2.0**
3	9.2	6.5	**2.0**	7.0	*5.0*	**2.0**	**2.0**
4	10.9	13.8	16.0	18.0	11.0	*8.0*	**7.0**
5	9.3	10.1	5.0	7.0	*4.0*	**3.0**	*4.0*
6	6.5	5.3	6.0	**2.0**	8.0	*4.0*	*4.0*
7	3.4	4.9	*2.0*	**1.0**	5.0	**1.0**	**1.0**
8	19.5	15.3	17.0	18.0	8.0	*6.0*	**4.0**
9	6.3	6.8	4.0	**2.0**	5.0	*3.0*	*3.0*
10	7.2	8.8	7.0	10.0	*4.0*	**2.0**	**2.0**
11	8.9	9.9	6.0	**4.0**	9.0	*5.0*	6.0
12	22.0	26.4	26.0	19.0	23.0	*14.0*	**13.0**
13	76.6	79.1	**35.0**	**35.0**	68.0	*54.0*	60.0
14	10.0	12.2	14.0	10.0	*6.0*	**5.0**	*6.0*
15	4.0	3.3	**2.0**	*3.0*	*3.0*	**2.0**	**2.0**
16	27.1	22.2	28.0	33.0	31.0	*20.0*	**14.0**
17	4.9	4.5	*4.0*	*4.0*	6.0	**3.0**	**3.0**
18	13.7	17.2	23.0	25.0	*10.0*	**9.0**	*10.0*
19	24.9	20.9	31.0	**11.0**	25.0	16.0	*13.0*
20	13.6	19.0	**4.0**	10.0	30.0	11.0	*9.0*
21	10.2	9.5	10.0	10.0	12.0	*8.0*	**7.0**
22	26.7	22.6	**13.0**	34.0	*14.0*	*14.0*	16.0
23	17.4	12.7	9.0	10.0	*4.0*	**2.0**	**2.0**
24	5.3	6.9	*5.0*	**3.0**	6.0	**3.0**	**3.0**
25	29.4	32.4	20.0	*16.0*	42.0	18.0	**14.0**
	0/*0*	**0**/*0*	**5**/*4*	**8**/*4*	**0**/*8*	**10**/*12*	**14**/*7*

Table 7: The average CPU time (ms).

ID	Random	Kpp	PCA-Part	Var-Part	Maximin	AIF-SLI-t-NN	AIF-SLI-ϵ-Balls
1	155	144	158	88	166	1485	567
4	7	9	8	10	6	14	10
5	5	6	5	5	4	9	7
8	259	323	515	320	221	501	326
13	894	1592	602	568	1315	15497	4712
18	187	326	830	487	195	470	169
22	259	323	515	320	221	501	326
25	19	26	23	16	34	171	91

Table 8: CPU time (ms) of each component in AIF-SLI-t-NN.

ID	AIF-SLI-t-NN	Calculated $\vec{d}$	Initialization	run K-means
1	1612	1206	19	215
4	25	6	4	12
5	22	5	5	7
13	15666	13565	46	1368
18	420	235	62	109
22	485	251	31	165
25	207	142	9	15

which explains the reason why our method can reduce the iteration significantly.

8. Conclusion

In this paper, we propose an adaptive initialization framework based on spatial local information (AIF-SLI) for k-means algorithm, which is designed by taking advantage of the local density of data points. We first describe the framework of AIF-SLI based on defining a function to describe local density for data points. Since it is difficult to estimate the local density correctly, we derive two approximate estimations: density by t-nearest neighborhoods (t-NN) and density by ϵ-neighborhoods (ϵ-Ball), leading to two implements of the proposed framework. Experiments on more than 20 datasets show promising performance of the proposed methods and denote that the proposed methods have several advantages: (1) they can find the reasonable candidates of initial centers effective; (2) they are robust to outliers; (3) they can reduce the iterations of k-means methods significantly; and (4) they are easy to implement.

In the future, taking the Gaussian Mixture Model (GMM), for example, we plan to extend our framework to other clustering algorithms. And for the rapid growth of applications for Big Data, we also plan to parallel our framework on GPU or MapReduce platform to accelerate k-means algorithm for large scale applications.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to thank the referees for the very helpful remarks which helped improve the paper. This work is supported by the National Natural Science Foundation of China (no. 61300140).

References

[1] J. A. Hartigan, *Cluster Analysis*, John Wiley & Sons, 1975.

[2] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[3] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 316–323, 1999.

[4] K. Grauman and T. Darrell, "The pyramid match kernel: discriminative classification with sets of image features," in *Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV '05)*, vol. 2, pp. 1458–1465, IEEE, October 2005.

[5] M. E. Celebi, "Improving the performance of k-means for color quantization," *Image and Vision Computing*, vol. 29, no. 4, pp. 260–271, 2011.

[6] B. Liu, X. Li, W. S. Lee, and P. S. Yu, "Text classification by labeling words," in *Proceedings of the Innovative Applications of Artificial Intelligence Conference (IAAI '04)*, pp. 425–430.

[7] A. Coates and A. Y. Ng, "Learning feature representations with k-means," in *Neural Networks: Tricks of the Trade*, pp. 561–580, Springer, 2012.

[8] X. Wu, V. Kumar, Q. J. Ross et al., "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.

[9] C.-Y. Tsai and C.-C. Chiu, "Developing a feature weight self-adjustment mechanism for a K-means clustering algorithm," *Computational Statistics and Data Analysis*, vol. 52, no. 10, pp. 4658–4672, 2008.

[10] 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*, vol. 40, no. 1, pp. 200–210, 2013.

[11] M. E. Celebi and H. A. Kingravi, "Deterministic initialization of the k-means algorithm using hierarchical clustering," *International Journal of Pattern Recognition and Artiffcial Intelligence*, vol. 26, Article ID 1250018, 25 pages, 2012.

[12] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, Berkeley, Calif, USA, 1967.

[13] E. W. Forgy, "Cluster analysis of multivariate data: efficiency vs interpretability of classifications," *Biometrics*, vol. 21, pp. 768–769, 1965.

[14] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proceedings of the 18nth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035, Society for Industrial and Applied Mathematics, 2007.

[15] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable kmeans++," *Proceedings of the VLDB Endowment*, vol. 5, no. 7, pp. 622–633, 2012.

[16] G. H. Ball and D. J. Hall, "A clustering technique for summarizing multivariate data," *Behavioral Science*, vol. 12, no. 2, pp. 153–155, 1967.

[17] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.

[18] I. Katsavounidis, C.-C. J. Kuo, and Z. Zhang, "New initialization technique for generalized Lloyd iteration," *IEEE Signal Processing Letters*, vol. 1, no. 10, pp. 144–146, 1994.

[19] M. Erisoglu, N. Calis, and S. Sakallioglu, "A new algorithm for initial cluster centers in k-means algorithm," *Pattern Recognition Letters*, vol. 32, no. 14, pp. 1701–1705, 2011.

[20] T. Su and J. G. Dy, "In search of deterministic methods for initializing K-means and Gaussian mixture clustering," *Intelligent Data Analysis*, vol. 11, no. 4, pp. 319–338, 2007.

[21] T. Onoda, M. Sakai, and S. Yamada, "Careful seeding method based on independent components analysis for k-means clustering," *Journal of Emerging Technologies in Web Intelligence*, vol. 4, no. 1, pp. 51–59, 2012.

[22] M. B. Al-Daoud and S. A. Roberts, "New methods for the initialisation of clusters," *Pattern Recognition Letters*, vol. 17, no. 5, pp. 451–455, 1996.

[23] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[24] K. Bache and M. Lichman, UCI machine learning repository, 2013, http://archive.ics.uci.edu/ml.

[25] G. W. Milligan and M. C. Cooper, "A study of standardization of variables in cluster analysis," *Journal of Classification*, vol. 5, no. 2, pp. 181–204, 1988.