Neurocomputing 143 (2014) 109–122

Contents lists available at ScienceDirect

Neurocomputing

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

A novel graph-based *k*-means for nonlinear manifold clustering and representative selection

Enmei Tu [a], Longbing Cao [b], Jie Yang [a,*], Nicola Kasabov [c]

[a] Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, China
[b] UTS Advanced Analytics Institute, University of Technology Sydney, Australia
[c] The Knowledge Engineering and Discovery Research Institute, Auckland University of Technology, New Zealand

ARTICLE INFO

Article history:
Received 6 November 2013
Received in revised form
4 April 2014
Accepted 20 May 2014
Communicated by Liang Wang
Available online 20 June 2014

Keywords:
k-means
Manifold clustering
Random walk
Graph learning

ABSTRACT

Many real-world applications expose the nonlinear manifold structure of the lower dimension rather than its high-dimensional input space. This greatly challenges most existing clustering and representative selection algorithms which do not take the manifold characteristics into consideration. The performance of the corresponding learning algorithms can be greatly improved if the manifold structure is considered. In this paper, we propose a graph-based *k*-means algorithm, GKM, which bears the simplicity of classic *k*-means while incorporating global information of data geometric distribution. GKM fully exploits the intrinsic manifold structure for appropriate data clustering and representative selection. GKM is evaluated on both synthetic and real-life data sets and achieves very impressive results compared to the state-of-the-art approaches, including classic *k*-means, kernel *k*-means, spectral clustering, and clustering through ranking and for representative selection. Given the widespread appearance of manifold structures in real world problems, GKM shows promising potential for partitioning manifold-distributed data.

1. Introduction

Clustering is aimed to divide a set of samples $\mathcal{X}=\{x_1,x_2,\dots,x_n\}$ into K disjointed subsets $\mathcal{X}_1,\mathcal{X}_2,\dots,\mathcal{X}_K$ so that points in the same subset share common properties while points which belong to different subsets do not share these properties. There are many algorithms available for performing this task, with *k*-means [1] being the most popular because of its properties as follows: it can effectively partition data with Gaussian-like distributions; it is intuitive and easy to implement; its implementation is of linear computational complexity, etc. However, *k*-means also exhibits typical disadvantages for manifold distributed data sets and our motivation for extending *k*-means to a manifold algorithm is based on the following observations. (1) Recently, it was widely acknowledged in the data analysis and machine learning communities that many real world data sets, such as face images, voice spectrum, hand-writing digital images and document contents, stringently obey the low-rank rules, which means that they are distributed on a manifold whose dimensionality is often much lower than the ambient space [2,3]. The classic *k*-means algorithm, CKM, does not take this important characteristic into account and thus performs poorly when dealing with such data sets. (2) The cluster centers (or prototypes) given by CKM are generally not members of a data set and thus cannot be used directly for many applications, such as video and text summarization [4,5], which aim to choose a small subset of frames or sentences that best describe the overall contents. Compared to CKM, the kernel *k*-means [6], KKM, in most cases, yields better results by mapping samples to a possibly much lower dimensional space than that in the input space, in which the manifolds of different classes are separable. Unfortunately, such a perfect mapping may not exist in practice, nor is it clear what kinds of mapping exist for a given data set.

Inspired by the isometric feature mapping (isomap)-based nonlinear dimension reduction algorithm [3] and the manifold ranking algorithm [7], we propose a graph-based *k*-means algorithm, GKM, to overcome the above disadvantages by taking the geometric characteristics of data distribution into account. Particularly, GKM can fully exploit the underlying manifold structure of a data set to produce better clustering results and, meanwhile, identify a suitable data point representative (or centroid) for each subset. GKM also retains the advantages of CKM, such as ease of implementation, intuitive and low computational complexity. Extensive experiments conducted on both synthetic and real-world data sets validate that GKM is very effective for manifold clustering with appropriate representative selection.

[*] Corresponding author.
E-mail address: jieyang@sjtu.edu.cn (J. Yang).

http://dx.doi.org/10.1016/j.neucom.2014.05.067
0925-2312/© 2014 Elsevier B.V. All rights reserved.

The remainder of this paper is organized as follows: Section 2 reviews the related works in nonlinear manifolds clustering and representative selection and Section 3 briefly reviews the classic k-means algorithm. Section 4 presents our graph-based k-means algorithm. Finally, the simulations and comparisons are presented in Section 5, followed by discussions and conclusions in Section 6.

2. Related works

Nonlinear manifold clustering and representative selection are very challenging topics in machine learning. The work in [8,9] extends mean shift to nonlinear manifold clustering and performs well in motion segmentation, but it is restricted to analytic manifolds. Goh and Vidal [10] only consider separated nonlinear manifold clustering. In recent years, spectral clustering [11–15] and manifold clustering algorithms [16–18] have been proposed to handle general manifold-distributed data sets. These approaches either do not present a meaningful subset of data points which can mostly represent the data set or need a complicated optimization process. Other algorithms have been specially designed for the selection of representative data points [5,4] on the condition of solving an unsmooth optimization problem, thus suffering from high computational costs in dealing with large data sets. In addition, they also do not present a clear description of data clusters and thus cannot tell which points are best represented by which representative. For other similar data-driven researches and applications, we recommend the recent results in [19,20]. Very limited work has been conducted on k-means for manifolds, and as yet, this area has not been fully exploited. The work in [21] is restricted to data on sphere and [22] aims to analyze the reconstruction error and learning rate of k-means on manifolds but does not provide a concrete algorithm for clustering.

There are also studies of k-means on graph. In [23] Euclidean distance and centroid are replaced by graph edit distance and the so-called mean graph respectively, but the computational cost of computing both graph edit distance and mean graph are very high and thus make the algorithm not suitable for large data sets. In [24] the classic k-means is just utilized as a post-processing method after thresholding the sequence of edge lengths that added to the minimal spanning tree by Prims algorithm to obtain the final clusters.

In contrast, in this paper we make two essential changes to the classic k-means for dealing with nonlinear manifold data. Particularly, we first extend the centroid concept of point cloud in Euclidean geometry to the centroid of manifold in Riemannian geometry. Then, borrowing from graph-based semi-supervised learning method, a new random walk model, the tired random walk model which is capable of describing the similarity between points on nonlinear manifold, is proposed to determine the centroid–member relationships on graph.

3. Review of the classic *k*-means clustering algorithm

Let $\mathcal{X}=\{x_1,x_2,\dots,x_n\}\in\mathbb{R}^d$ be a sample set to be partitioned into K groups and $\mathcal{Y}=(y_1,y_2,\dots,y_n)$, $y_i\in\{1,2,\dots,K\}$ be the label vector in which each component gives the class label of the corresponding sample in $\mathcal{X}$. Clustering is aimed to partition $\mathcal{X}$ into K disjointed subsets $\mathcal{X}_1,\mathcal{X}_2,\dots,\mathcal{X}_K$ so that samples in the same subset have same class labels, i.e. $y_i=k$, $x_i\in\mathcal{X}_k$, $k\in\{1,\dots,K\}$. We denote by $\mathcal{C}_k=\{i_1,i_2,\dots,i_{|\mathcal{C}_k|}\}$ the index set of elements in subset $\mathcal{X}_k$.

In classic k-means (CKM), the algorithm updates iteratively the cluster centroids $\{c_1,c_2,\dots,c_K\}$ and index sets $\mathcal{C}_k, k=1..K$. For a particular cluster, the centroid of the cluster is updated by simply averaging the memberships over its members, and the membership of a sample is determined by the nearest Euclidean distance from it to all the centroids. Mathematically, the algorithm updates the cluster centroid by

$$c_k=\frac{1}{|\mathcal{C}_k|}\sum_{i\in\mathcal{C}_k}x_i \tag{1}$$

After all centroids are updated, the algorithm re-computes the Euclidean distance $d_E(x,y)=\|x-y\|^2$ between each sample and all the cluster centroids and labels the sample as a member of the cluster whose centroid achieves the smallest Euclidean distance

$$y_i=\arg\min_{k=1\dots K} d_E(x_i,c_k) \tag{2}$$

In the first iteration, the algorithm randomly chooses K points $\{c_1,c_2,\dots,c_K\}$ to be the initial centroids. Thereafter, these two steps iterate alternatively until the algorithm converges.

4. A graph *k*-means manifold clustering algorithm

In this section, we propose a graph-based k-means algorithm, GKM, which takes the intrinsic manifold structure into account. Similar to classic k-means, GKM also has two essential steps, updating centroids and updating the samples membership, but these steps differ in nature from their counterparts in the classic k-means. We also present an initialization method to obtain a group of high quality initial centroids. These are described in detail as follows.

4.1. *Updating centroids*

In the classic k-means, the k-th centroid obtained by Eq. (1) is the coordinates mean of the point cloud. This is the classic centroid concept in Euclidean geometry. Here we extend this concept to Riemannian geometry, *i.e.* computing the centroid of a manifold. Note that the centroid in Eq. (1) is actually the optimal solution of the following optimization problem:

$$c_k=\arg\min_{x\in\mathbb{R}^d}\frac{1}{2}\sum_{i\in\mathcal{C}_k}d_E(x,x_i),\quad k=1\dots K \tag{3}$$

where $d_E(x,x_i)=\|x-x_i\|^2$ is the Euclidean distance. For data points sampled from manifolds, this optimization has two defects: (a) the centroid may move off the manifold and thus it cannot represent the underlying data distribution well; (b) the Euclidean distance cannot reflect the true relationships between samples in Riemannian geometry because it does not fully capture the data geometric feature and thus it is not a proper measure of similarity.

To capture the intrinsic geometric feature, we generalize problem (3) by (a) restricting the centroid on the manifold[1]; and (b) using geodesic distance as a measurement between two points. Combining these two we extend the classic centroid in Euclidean geometry to the manifold centroid in Riemannian geometry by solving the following optimization problem:

$$c_k=\arg\min_{x_j,j\in\mathcal{C}_k}\frac{1}{2}\sum_{i\in\mathcal{C}_k}d_g(x_j,x_i),\quad k=1\dots K \tag{4}$$

where $d_g(x_i,x_j)$ is the geodesic distance between two samples x_i and x_j. However, in clustering settings, the exact geodesic distance between two samples x_i and x_j usually cannot be obtained directly, because we have no prior information about the underlying

[1] To restrict centroid on manifold, it is also possible to adopt other method to choose the class centroid, such as the medium point. But our optimization method has at least two benefits: first, the formulation is straightway to classic k-means practitioners and easy to be understood; second, it is of computational efficiency, as will be demonstrated.

manifolds. However, as pointed out in [3], if a data set has sufficient points sampled from the manifold, then the graph distance will be a good approximation of the geodesic distance.

Given a graph, the graph distance between two vertices x_i and x_j is defined as the shortest path of all paths connecting them. Therefore, if we construct a graph on data set $\mathcal{X}$, we can alternatively optimize the following problem to determine the centroids:

$$c_k = \arg\min_{x_j, j \in \mathcal{C}_k} \frac{1}{2} \sum_{i \in \mathcal{C}_k} d_G(x_j, x_i), \quad k = 1 \ldots K \tag{5}$$

where $d_G(x_i, x_j)$ denotes the graph distance between vertices x_i and x_j. As the centroids are members of the vertex set, the optimization problem can be easily solved with the graph distance matrix given. Specifically, given the pairwise graph distance matrix $D_G = \{d_G(x_i, x_j) | x_i, x_j \in \mathcal{X}\}$, the graph distance matrix corresponding to cluster $\mathcal{X}_k$ is simply the principal submatrix of D_G, i.e. $D_G^k = \{d_G(x_i, x_j) | i, j \in \mathcal{C}_k\}$. The solution to problem (5) is the vertex corresponding to the minimal row sum of

$$c_k = x_j; \quad j = \arg\min_{i = 1 \ldots |\mathcal{C}_k|} [D_G^k e]_i, j \in \mathcal{C}_k \tag{6}$$

where $e = (1, 1, \ldots, 1)^T$. $D_G^k e$ computes the row sum of submatrix D_G^k and j is the optimal index corresponding to the minimal row sum. Now we simply compute the graph distance matrix and repeatedly compare its principal submatrix row sum to determine the clusters centroids. The method for obtaining D_G will be presented in Section 4.4.

To illustrate the difference between results of Eqs. (1) and (6), let us consider an example of two one-dimension manifolds in Fig. 1(a), the outer arch manifold and the inner reflected S manifold. The triangles on outer manifold belong to class one and the squares on inner manifold belong to class two.

The red triangle and green square in Fig. 1(b) are the classic centroids of class one and class two, respectively, obtained with Eq. (1). The color shapes in Fig. 1(c) are the respective manifold centroids obtained with Eq. (6). From Fig. 1(b) we can see that the classic centroids provide us little useful information about the manifold distribution, because one can hardly make correct judgment which manifold the samples belong to by comparing their distances to the two centroids. But in Fig. 1(c) the centroids are located correctly on their own manifold and thus can represent the underlying data distribution very well. It is also interesting to note that for symmetrical manifold (i.e., the reflected S), there is little difference between classic centroid and manifold centroid. But for unsymmetrical manifold (i.e., the outside arch), manifold centroid is more effective to describe the distribution. Moreover, by choosing a proper similarity measure, the samples on different manifolds can be expected to be distinguished easily. In the following section, we will describe a new random walk model that can better describe the similarity on manifold.

4.2. Updating memberships

The classic k-means updates memberships according to the shortest Euclidean distance rule, which is not a proper measure on manifold. Here we introduce a tired random walk model to describe the centroid–member relationship between points on a manifold. While the matrix expression of the tired random walk model appeared in [25,26], we have not seen a model in the literature that endows it with a clear physical meaning and interprets it for the purpose of similarity measure on manifolds. Our model works as follows.

Recall that the random walk transition matrix is $P = D^{-1}W$ and the t-step transition matrix is P^t, where W is the graph adjacency matrix and D is a diagonal matrix with entries $D_{ii} = \sum_{j=1}^{n} W_{ij}$, which is the degree of vertex x_i. Imagine that a tired random walker, who walks continuously through edges in a graph, becomes more tired after each walk and finally stops after all energy is exhausted. Mathematically, the transition probability of strength reduces after each walk and finally reaches 0. Assume that the rate of strength reduction is $\alpha \in (0, 1)$, then after t-steps, the transition matrix becomes $(\alpha P)^t$. Now suppose that the random walker starts from vertex x_i and its destination is vertex x_j, then it may walk through any path that connects x_i and x_j with arbitrary steps. As a result, the accumulated transition probability for all possible paths is $\overline{P}_{ij} = [\sum_{t=0}^{\infty} (\alpha P)^t]_{ij}$. This accumulated transition probability measures the ability of a random walk starting from x_i being able to reach x_j before its strength is used up. It takes all the possible paths into consideration and thus captures the global geometry of the underlying manifolds. In a matrix form, the corresponding accumulated transition probability matrix is $\overline{P} = \sum_{t=0}^{\infty} (\alpha P)^t$. As the eigenvalue of P is in $[-1, 1]$ and $\alpha \in (0, 1)$, the series converges to $\overline{P} = (I - \alpha P)^{-1}$, where I is the identity matrix. Moreover, the manifoldranking matrix is $R = (I - \alpha S)^{-1}$ [7], where $S = D^{-1/2} W D^{-1/2} = D^{1/2} P D^{-1/2}$ is the normalized adjacency matrix. Therefore

$$\overline{P} = D^{-1/2} R D^{1/2} \tag{7}$$

or entry-wise

$$\overline{P}_{ij} = \sqrt{\frac{D_{jj}}{D_{ii}}} R_{ij} \tag{8}$$

Columns of the ranking matrix encode the information spreading from a seeding vertex to all other vertices in graph through edges, so α can also be interpreted as information loss on the edges during spreading. The ranking matrix R is symmetric, so any two vertices are equal to each other and one has no priority over the

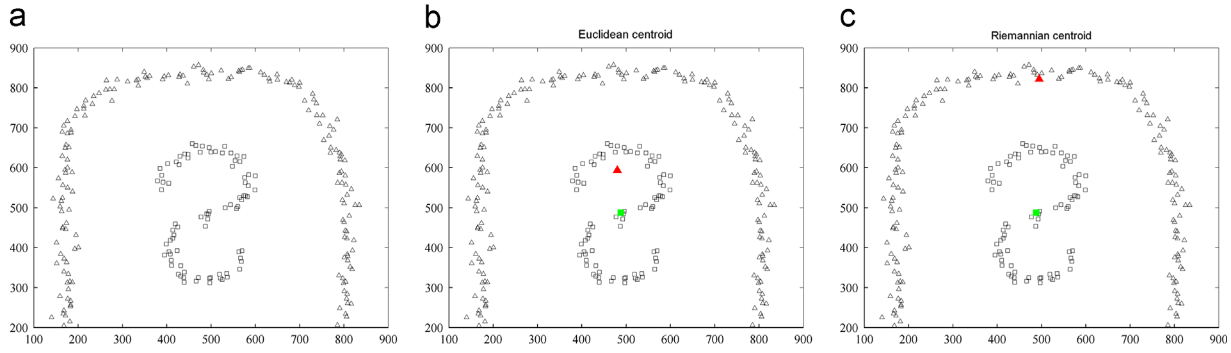


Fig. 1. A toy example data set. (a) The data set contains two one-dimension manifolds, an arch outside labeled by triangles and a reflected S inside labeled by squares. (b) The classic centroids found by Eq. (1). (c) The manifold centroids found by Eq. (6). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

other. By contrast, from Eq. (7) or (8) we know that if the degree of vertex x_j is larger than the degree of vertex x_i (i.e. vertex x_j is in a denser region[2] than vertex x_i), $\overline{P}_{ij}$ will be larger than $\overline{P}_{ji}$. This means that the probability of a random walker moving from vertex x_i to vertex x_j (i.e. from lower density region to higher density region) is larger than moving from vertex x_j to vertex x_i (i.e. from higher density region to lower density region). As a result, the memberships, and thus the centroids, tend to move to high density regions during iterations. This is the key reason we use $\overline{P}$ but not R. Note that for a constructed graph, matrix $\overline{P}$ keeps constant during iterations and thus it only needs to be computed once during iterations.

Finally, given centroids $x_{i_1}, x_{i_2}, \ldots, x_{i_K}$, we choose the columns of $\overline{P}$ that correspond to the centroids to form a matrix $\hat{P} \in \mathbb{R}^{n\times K}$. The membership of x_i is determined by

$$k = \arg \max_{j=1..K} \hat{P}_{ij} \tag{9}$$

The whole algorithm procedure is described in Table 1.

4.3. Removal of bridging points

In real world applications, the gap between different manifolds is usually not very clear due to noise contamination or closeness of classes. Here we propose a simple but effective method, message routing, to remove the bridging points between manifolds. Suppose a message can be transferred on the graph from one vertex to another through edges connecting them. Let $q_i, i=1,2,\ldots,n$ denote the message amount processed by each vertex i. We first randomly choose two vertices and send a message from one (any one, since this is a undirect graph) to the other via the graph distance path. If vertex i is in the path, q_i increases 1. These steps repeat Q times, where Q is a user set value. After this, we compute $\overline{q_i} = q_i/\sqrt{D_{ii}}, i=1,2,\ldots,n$ and sort $\overline{q_i}$ in a descend order. The vertices corresponding to the first N_b values are identified as the bridging points. Because graph distance path has been obtained simultaneously with little efforts while computing graph distance matrix, this process is very time efficient. Fig. 2 shows the results with $Q=1000$ and $N_b=30$ on a noisy two-circle data set.

4.4. Large data set considerations

The computational cost for updating centroids and memberships is low because these steps only compare elements of submatrix entries. In addition, experiments show that the algorithm generally converges after very few steps. For large data sets, the main computational cost lies in computing D_G and $\overline{P}$. A straightforward computation needs $O(n^3)$ operations for each of them. However, both of the two matrices can be computed approximately with sufficient accuracy in a very efficient way. Specifically, we compute D_G by adopting the Dijkstra's algorithm [27] because manifold-distributed data points often form a very sparse graph. The sparsity of the neighborhood graph in computing the shortest path can be fully exploited by the Dijkstra's algorithm. For matrix $\overline{P}$, we first obtain matrix R using the Nyström method [28] and then use Eq. (7) or (8) to obtain $\overline{P}$. Particularly, if the graph adjacency matrix is positive semi-definite, we can apply the Nyström method to S directly; otherwise, a jittering factor can be added to S to keep it positive semi-definite. Note that matrix R can be written as $R=((1-\alpha)I+\alpha\mathcal{L})^{-1}$, where $\mathcal{L}=I-S$ is the normalized graph Laplacian and it is always positive semi-definite, thus we can alternatively apply the Nyström method to $\mathcal{L}$. After this, the Woodbury formula is used to compute the inversion efficiently. The time complexity of Dijkstra's algorithm is $O(Nn+n \log n)$, where N is the number of the nearest neighbors in a k NN graph. The complexity of both the Nyström method and the Woodbury formula is $O(m^3+mn)$, where $m \ll n$ is the number of rows/columns used to approximate the positive semi-definite. For large data sets, $n \log n$ becomes dominated and thus GKM scales linearithmically in terms of data set size n.

[2] Here by using *denser region* we mean that sample number in a unit volume is larger. In a k NN graph, a large vertex degree indicates that the neighbors of that vertex are very close to it, and thus the samples distribute densely in the region around the vertex.

Table 1
Algorithm procedure.

Steps	
1	Input data set $\mathcal{X}$ and parameters: K, N, σ
2	Initialize cluster centroids $c_1, c_2, \ldots, c_K$
3	Calculate graph distance matrix D_G
4	Calculate accumulated transition probability matrix $\overline{P}$
5	While centroids change
6	Update membership with Eq. (9)
7	Update centroids with Eq. (6)
8	End while
9	Output centroids $c_1, c_2, \ldots, c_K$ and label vector.

5. Experimental evaluation

5.1. Baseline algorithms

As the proposed algorithm is based on a graph, we call it Graph k-means (GKM),[3] which is compared to the classic k-means (CKM) [1] and kernel k-means (KKM) [6]. The recently proposed Clustering Through Ranking (CTR) algorithm [16] and Locally Consistent Concept Factorization (LCCF) [29] algorithm have been demonstrated to achieve state-of-art results for manifold clustering, thus we also choose them as baseline algorithms. Since spectral clustering algorithms achieve stable performance in most situations, we also include its normalized cut (NCut) edition [11] in the experiments for comparison. So the baseline algorithms are classic k-means (CKM),[4] kernel k-means (KKM),[5] Clustering Through Ranking (CTR),[6] Locally Consistent Concept Factorization (LCCF)[7] and normalized cut (NCut). The last three algorithms and the proposed GKM are all based on graph.

5.2. Experimental settings and parameters selection

For all graph-based algorithms (i.e. NCut, CTR, LCCF and GKM), we construct k NN graphs (in a k NN graph, a vertex only connects to its N nearest neighbors, and here N is chosen empirically from integers between 5 and 10 to produce good results since small N tends to perform well [30]), because k NN graph has been reported to produce better results for manifold-distributed data [31,2,3]. Moreover, adjacency matrix of a k NN graph is usually highly sparse and thus can be computed and stored in a very efficient way. The graph edges are weighted by Gaussian kernel function. Instead of tuning and choosing a fixed kernel width σ for the Gaussian kernel function, we treat σ as a independent variable and change it gradually from $0.01r$ to $100r$ with equally logarithmical interval to investigate the stability of clustering performance because σ is a key parameter in many algorithms that affects the

[3] Software is available upon request.
[4] We use the function *kmeans* in MATLAB 2013a.
[5] Software:http://www.kernel-methods.net/matlab_tools.html.
[6] Software:http://markus-breitenbach.com/machine_learning_code.php.
[7] Software:http://www.cad.zju.edu.cn/home/dengcai/Data/GNMF.html.

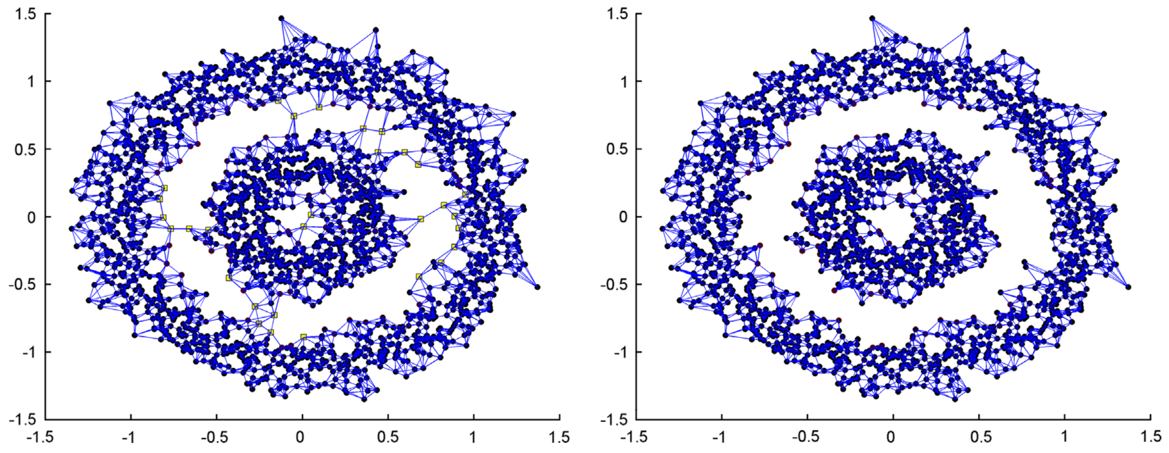


Fig. 2. Removal of bridging by message routing. Left: k NN graph before removing the bridging. Yellow squares are the bridging points identified by message routing method; Right: k NN graph after removing the bridging points. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

performance, where r is the mean distance from each sample to its N neighbors. For kernel k-means we also use Gaussian kernel to construct the kernel matrix and the kernel width is changed in the same way as in k NN graph. For NCut, we choose the bottom K (K is the class number of a data set) eigenvectors of $\mathcal{L}$ and perform a standard k-means on rows of the eigenvector matrix. For LCCF we use the default parameter values in the code (i.e. max iteration number 200, regularization parameter α 100, linear kernel and using NWC type weight), since according to the paper the performance are rather stable to these parameters. For GKM and CTR, the learning rate α is set to its typical value 0.99, as suggested by [25,16]. For GKM the neighborhood size parameter N for graph construction is also used in graph distance computation. Throughout all experiments, we use the original data directly and no special pre-processing technique or priori information about the databases are utilized. The cluster number is manually set for all the algorithms.

5.3. An initialization method

Given different initial conditions, the performance of k-means and its variants may vary greatly. Existing initialization methods include randomly sampling, uniformly sampling and subset pre-clustering. These methods do not take the whole data distribution geometry into consideration. Here we introduce a simple but effective initialization method. We first randomly choose K groups of centroids $\{c_1^j, c_2^j, \ldots, c_K^j\}, j=1\ldots K$. For each group, we then determine the corresponding sample label vector $\mathcal{Y}^i=(y_1^j, y_2^j, \ldots, y_n^j)$, $y_i^j \in \{1,2,\ldots,K\}, j=1..K, i=1\ldots n$ using the method described in Section 4.2. Thus for the K groups, we obtain K different initial label vectors $\mathcal{Y}^1, \mathcal{Y}^2, \ldots, \mathcal{Y}^K$ and each label vector gives a initial partitioning of the data set. We rank the K initial groups of centroid according to its maximal normalized associations (*knassoc*) [32] value and choose the group which has the largest *knassoc* value as the initial centroids. Since the graph adjacency matrix and the class indicator vector, which can be obtained from the label vector $\mathcal{Y}^k$ by setting the entries with one class label to 1 and all other entries to 0, are sparse, the *knassoc* can be evaluated with very little efforts.

5.4. Clustering performance

5.4.1. Assessment of clustering performance

We evaluate the clustering performance using two criteria. The first one, maximum bipartite matching, measures the clustering error rate. Specifically, assuming the clustering results are $y_{\tau_1}, y_{\tau_2}, y_{\tau_1}, \ldots, y_{\tau_n};\ y_{\tau_k} \in \{1,2,\ldots,K\}$ and the ground truth labels are $y_1, y_2, \ldots, y_n; y_k \in \{1,2,\ldots,K\}$ (where τ is an index permutation), then the matching degree of a permutation function τ is defined as

$$M_\tau = \sum_{k=1}^{n} I(y_k, y_{\tau_k}) \tag{10}$$

where $I(x,y)$ equals 1 if $x=y$, otherwise 0. Given a manifold-distributed data set, the ability of clustering algorithm $\mathcal{A}$ to detect the ground truth manifolds is measured by the maximum matching degree that can attain among all permutation functions. Thus, we define the associated error of clustering algorithm $\mathcal{A}$ as follows:

$$assoerr = \max_{\mathcal{P}} err(\tau) = \max_{\mathcal{P}} \left(1 - \frac{1}{n} M_\tau\right) \tag{11}$$

where $\mathcal{P}=\{\tau | \tau \leftarrow \mathcal{A}\}$ contains all the permutation produced by algorithm $\mathcal{A}$. In practice, enumerating all permutation functions to find the optimal one of Eq. (11) is almost impossible, but we can compare a T-suboptimal set and choose the best suboptimal one to approximate the optimal permutation function. A T-suboptimal set contains T suboptimal results given by clustering algorithm $\mathcal{A}$. As T approaches infinity, the best T-suboptimal set approaches a global optimal solution. In our experiments, we choose $T=20$ for each algorithm.

The second performance evaluation criterion is Normalized Mutual Information (NMI). The NMI is defined as

$$NMI(X,Y) = \frac{MI(X,Y)}{\sqrt{H(X)H(Y)}} \tag{12}$$

where $MI(X,Y)$ is the mutual information between two random variables X and Y, and $H(*)$ is the random variable entropy. Given ground-truth label vector and clustering result, the NMI measures how good the clustering results is, with respect to ground-truth. NMI$=1$ means the clustering result is perfect and NMI$=0$ means the clustering result is useless. Other value between 1 and 0 measures the quality of the clustering result.

5.4.2. Results of clustering synthetic data sets

To demonstrate the superiority of the proposed GKM over CKM and KKM for manifold distributed data clustering, we generate four synthetic data sets, shown in Fig. 3. The first two data sets are the two-moon and the two-circle data sets, which are widely used in semi-supervised learning researches to test the performance of manifold learning [25,33]. The third data set contains a ring of

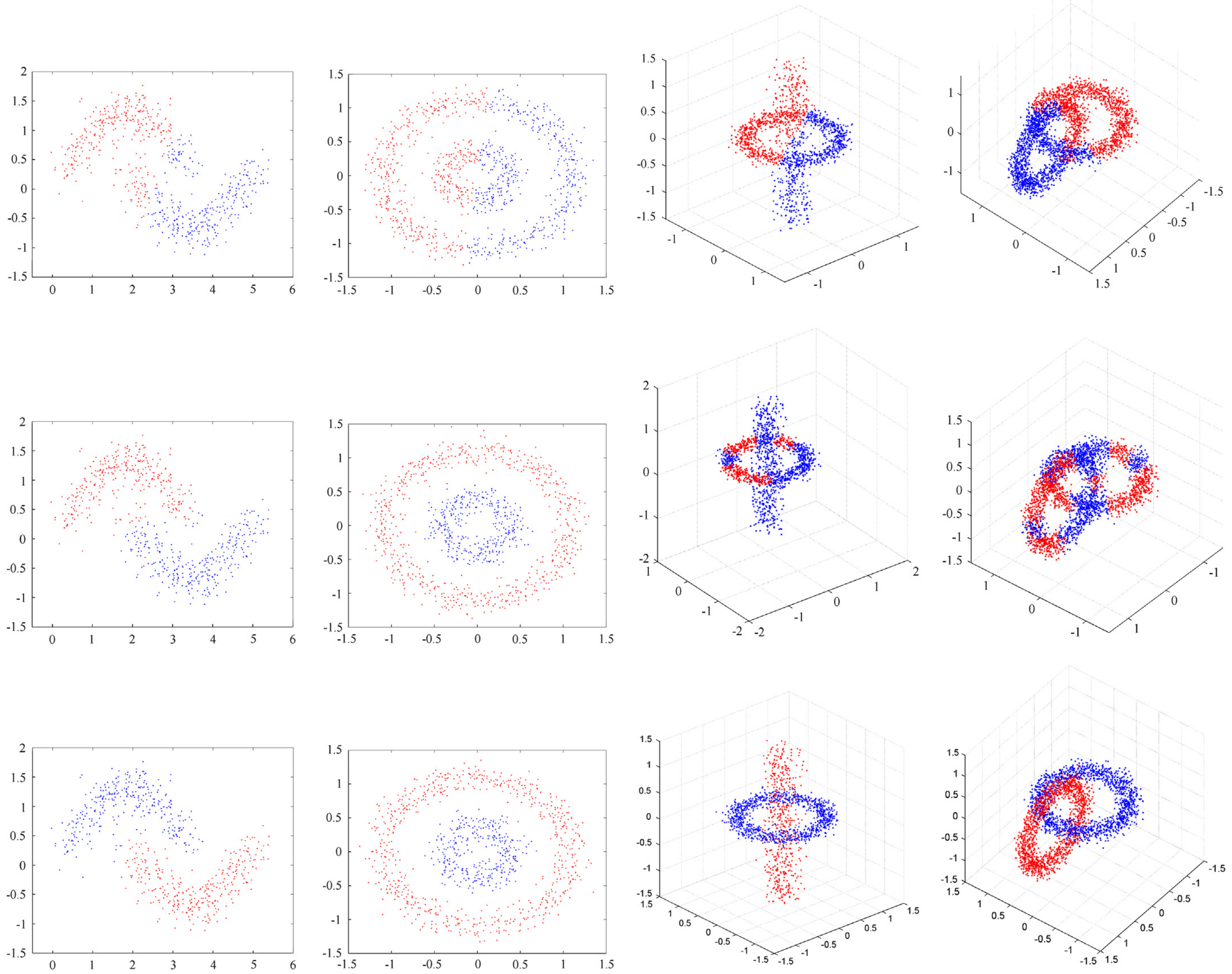


Fig. 3. Results of clustering synthetic manifold data sets. Note that unlike the toy data sets used in [16,34] where the data sets are fairly clean, the data sets used in this experiment are contaminated by a considerable level of noise and thus it is more difficult to cluster them correctly. (a) CKM, (b) KKM, and (c) GKM.

radius 0.8 and a cylinder of radius 0.2, and the fourth data set includes two knotted rings of radius 0.8. Both data sets are contaminated by Gaussian noise with variance 0.1. For KKM, the only parameter σ is searched over the range $[0.01r, 100r]$ and the value which yields the lowest error rate is chosen. For GKM we use $N=5$ and $\sigma=0.2r$. From these results we can see that CKM always partitions the data sets into half and half parts, regardless of the manifold structure. KKM can achieve better results than CKM on the first two data sets by mapping the input space to feature space, but it can hardly find out a good mapping for the last two data sets to achieve content results. For these four manifold distributed data sets, GKM can incorporate the information of the manifold structure and thus produces very good results. It is worth noting that these synthetic data sets are rather noisier than those in [16,34], and thus more challenging to be classified correctly.

We also conduct experiments to investigate the influence of data noise and kernel width upon the clustering performance. We run the algorithms with different kernel width and noise level and record the error rate for each run. The results are shown in Fig. 4. In each error rate map, the horizontal axis is the Gaussian kernel width σ, varying from $0.01r$ on the left to $100r$ on the right; the vertical axis is variance of Gaussian noise, varying from 0 at the top to 0.3 at the bottom. The color represents the clustering error rate, where dark blue equals 0 and dark red equals 0.5, as indicated by the color bar on the right of each map.

From these results, we can see that (a) CKM is not affected by kernel width[8] but its error rate is always around 0.2. (b) KKM has a very limited *Error Free Region* (EFR) (i.e. dark blue region) for small noise variance, which indicates that KKM is very sensitive to the kernel width. Even a small change in kernel width can cause the error rate increase very quickly. (c) LCCF is not so sensitive to σ but its EFR is very small. (d) CTR, NCut and GKM have very large and stable EFRs, but GKM has the smoothest and largest one among them, which indicates that GKM can endure a larger range of parameter change and noise contamination.

[8] Actually, the performance of CKM does not have any relationship with the kernel width because CKM does not use Gaussian kernel. But to be fair, we also run CKM at each time and record its performance and plot its error rate map for comparison.

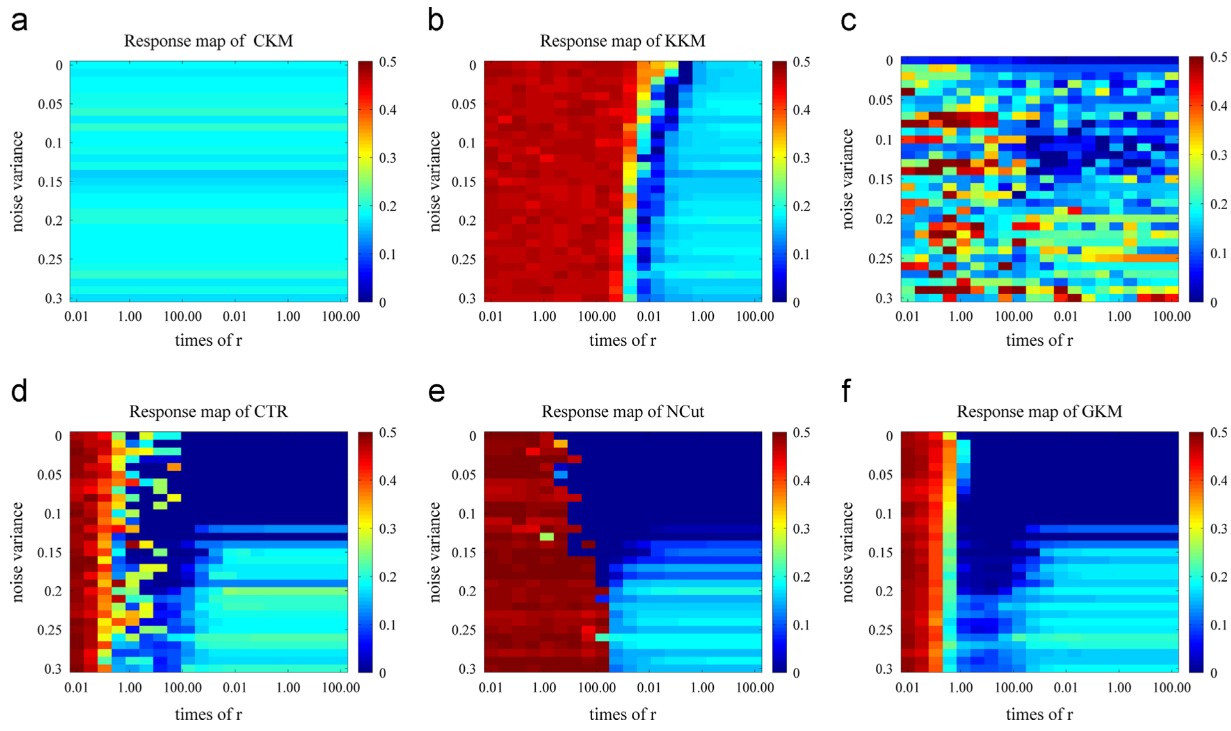


Fig. 4. Clustering error rate for different noise and kernel width on two-moon data set. (a) CKM, (b) KKM, (c) LCCF, (d) CTR, (e) NCut, and (f) GKM. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

5.4.3. *Results of clustering real-world data sets*

In this section we conduct experiments on three types of real-world data sets: face image, document content and other real world data sets from the widely used UC Irvine repository.

(1) *Face image clustering*

We carry out experiments on two widely used face image databases: the CMU PIE database[9] and the ORL database[10]. The PIE database contains 41,368 face images of 68 people. The size of each image is 64×64 pixels. For each person there are 13 different poses, 43 different illumination conditions and 4 different expressions. Because the whole database is very large, performing all the baseline algorithms on the whole database is very time and resource consumption. We conduct experiments on the Pose C07 subset.[11] The ORL database contains 400 face images from 40 persons, taken at different times and varying the lighting, facial expressions and facial details. The size of each image is 92×112 pixels. For the ORL we use the whole database. Fig. 5 displays some sample images of the two database.

Experimental results on the two face image databases are shown in Fig. 6. From these results we can see that CKM and KKM are generally perform worse than the four graph-based algorithms because they do not take the manifold structure into consideration. Among the four graph-based algorithms, the CTR tend to be more sensitive to kernel width than other three algorithms and GKM achieves the stablest and best results under both two performance evaluation criteria.

We also show in Fig. 7 the cluster centroid of the first two persons in each database found by GKM. Note how the centroids given by GKM represent the illumination and face position in each clusters. It is worth mentioning that this outcome is only given by GKM and other baseline algorithms cannot produce this result directly after clustering.

(2) *Text clustering*

Two popular text data sets are used to test the performance of the proposed GKM and the baseline algorithms: 20 newsgroups (version 2) and RCV1.[12] The 20 newsgroups data set contains 18,846 newsgroup documents with 26,214 features, partitioned (nearly) evenly across 20 different newsgroups. RCV1 data set contains 9625 documents from four categories with 29,992 distinct words. Again, because running the algorithms on these large data sets with large feature number is rather time and resource consumption, we use the *5% Training* subset, which contains 10 randomly selected subsets. For RCV1 we randomly divided the data set into subsets of size 500. Final results are averaged over the subsets.

Fig. 8 shows the experimental results. It is interesting that from these results we can see a similar "behavior" of CTR, NCut and LCCF, i.e. the similar trend of the performance curves against kernel width. The common point behind these three algorithms is that they all compute a low rank embedding using (or partially using) the graph Laplacian. So any local changes of graph weight matrix, hence the graph Laplacian, will exert a more or less similar impact on the embedding results and thus on the final performance. In contrast, GKM dose not have this similar behavior because it dose not depend on the eigen-spectrum of graph Laplacian. GKM uses the new tired random walk model, which takes all the possible path into consideration and thus can reflect more globally and stably the similarity on manifold, to determine the memberships and can achieve a better result.

[9] http://www.ri.cmu.edu/projects/project_418.html.
[10] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.
[11] http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html.
[12] Both can be downloaded from:http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html.

Fig. 5. Sample face images of CMU PIE database (left) and ORL database (right).

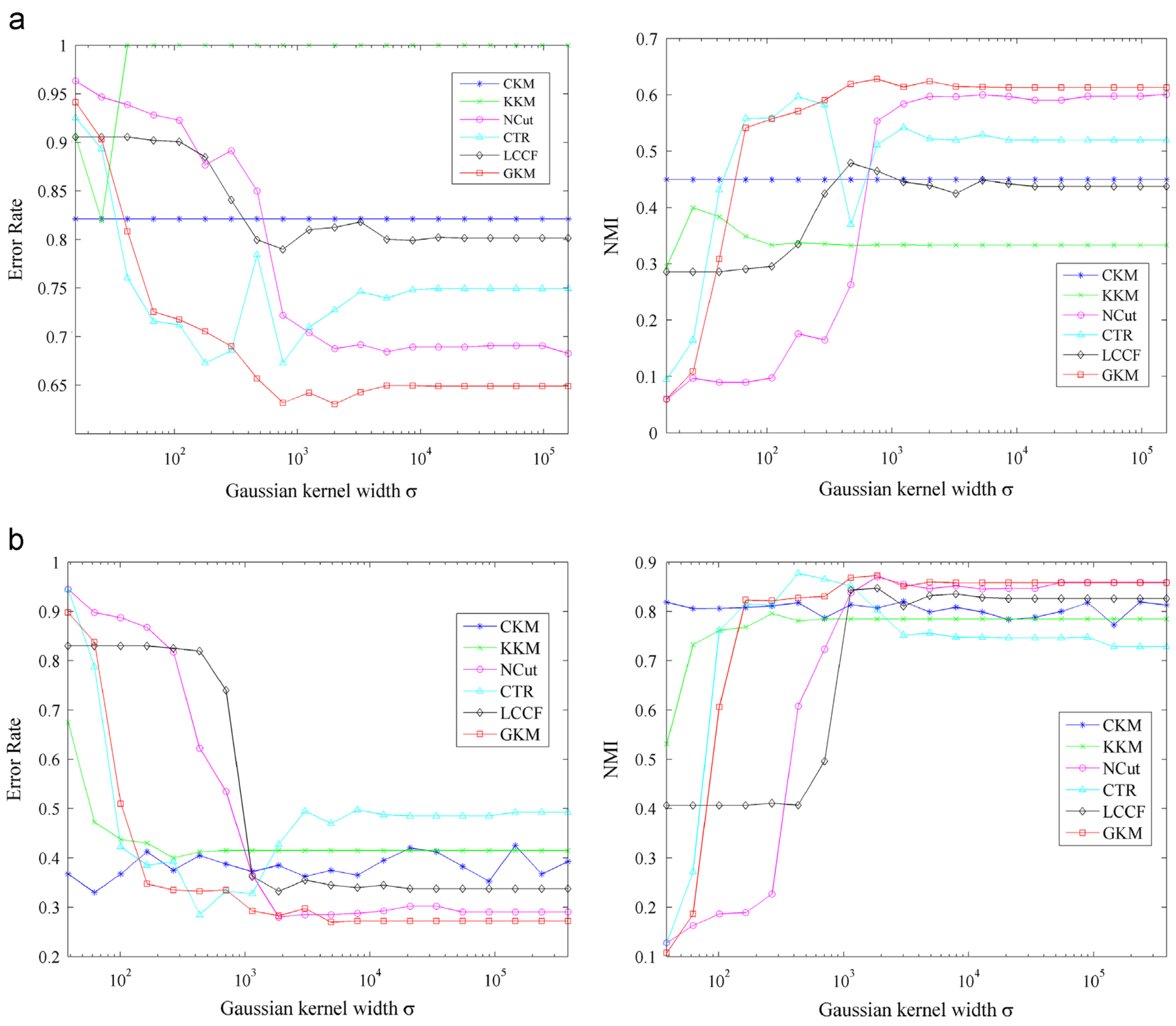


Fig. 6. Experimental results of (a) CMU PIE database and (b) ORL database.

(3) *UC Irvine repository*

UC Irvine repository[13] contains real-world data sets collected from various research fields. We conduct experiments on 4 numerical and no missing-value data sets from the repository: breasttissue (d:9, N:106, K:6)[14], vehicle (d:18, N:846, K:4), control data (d:6, N:600, K:6) and statlog (d:19, N:2310, K:7).

Fig. 9 displays the experimental results of the 4 data sets.

[13] http://archive.ics.uci.edu/ml/.

[14] d:feature dimension, N:sample number, K:class number.

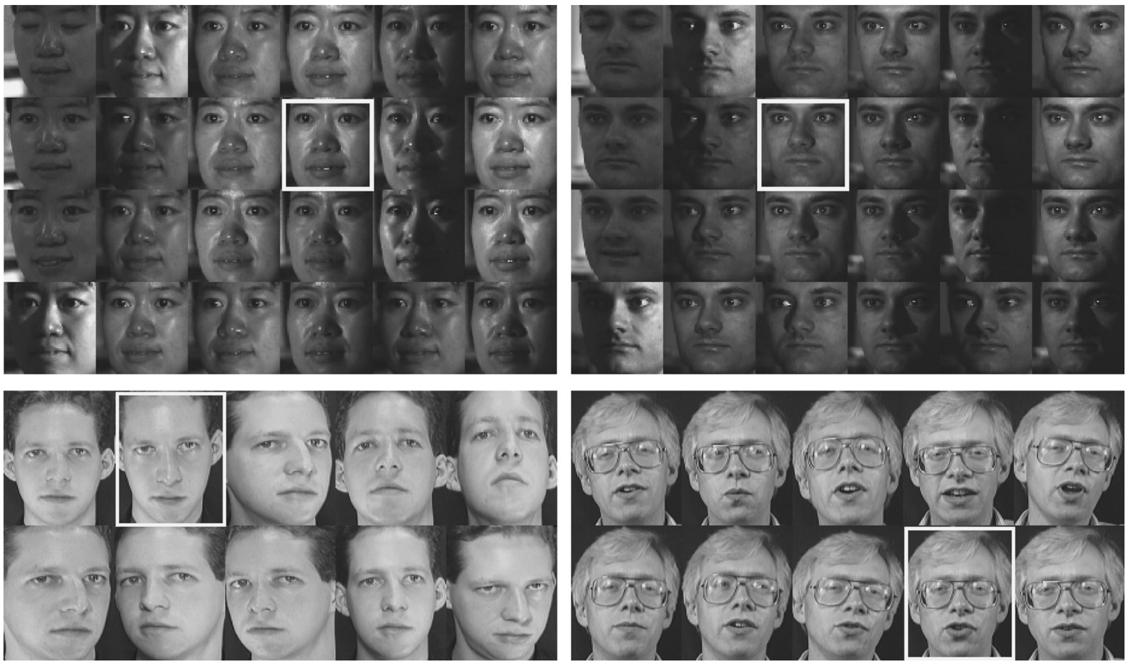
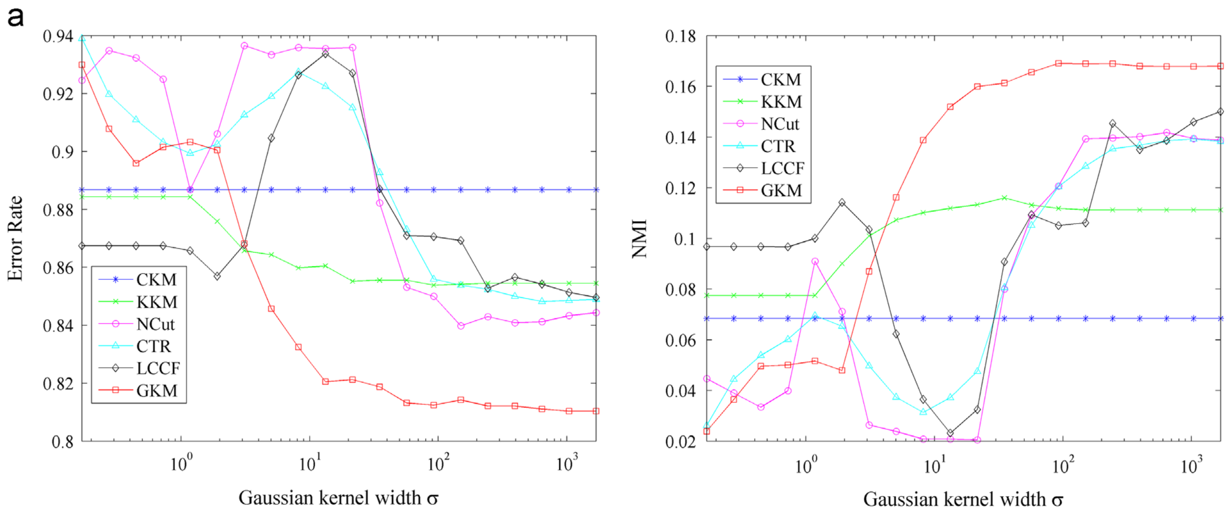


Fig. 7. Centroid of the first two persons' face images found by GKM. The images in white boxes are the centroid image. Top: CMU PIE database; Bottom: ORL database. Note that this is a unique property not shared by other baseline algorithms.



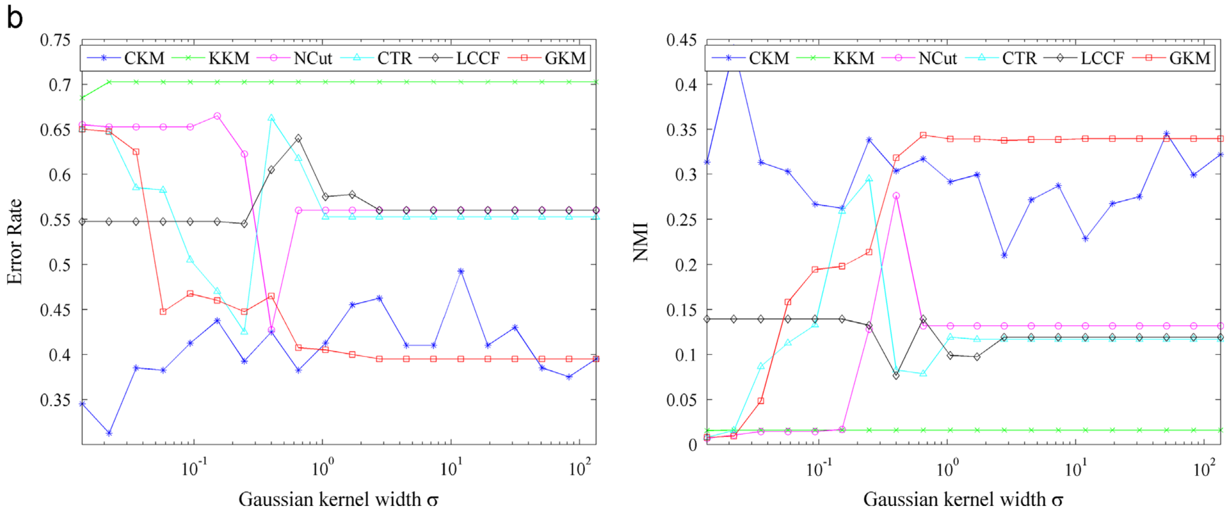


Fig. 8. Experimental results of (a) 20 newsgroups database and (b) RCV1 database.

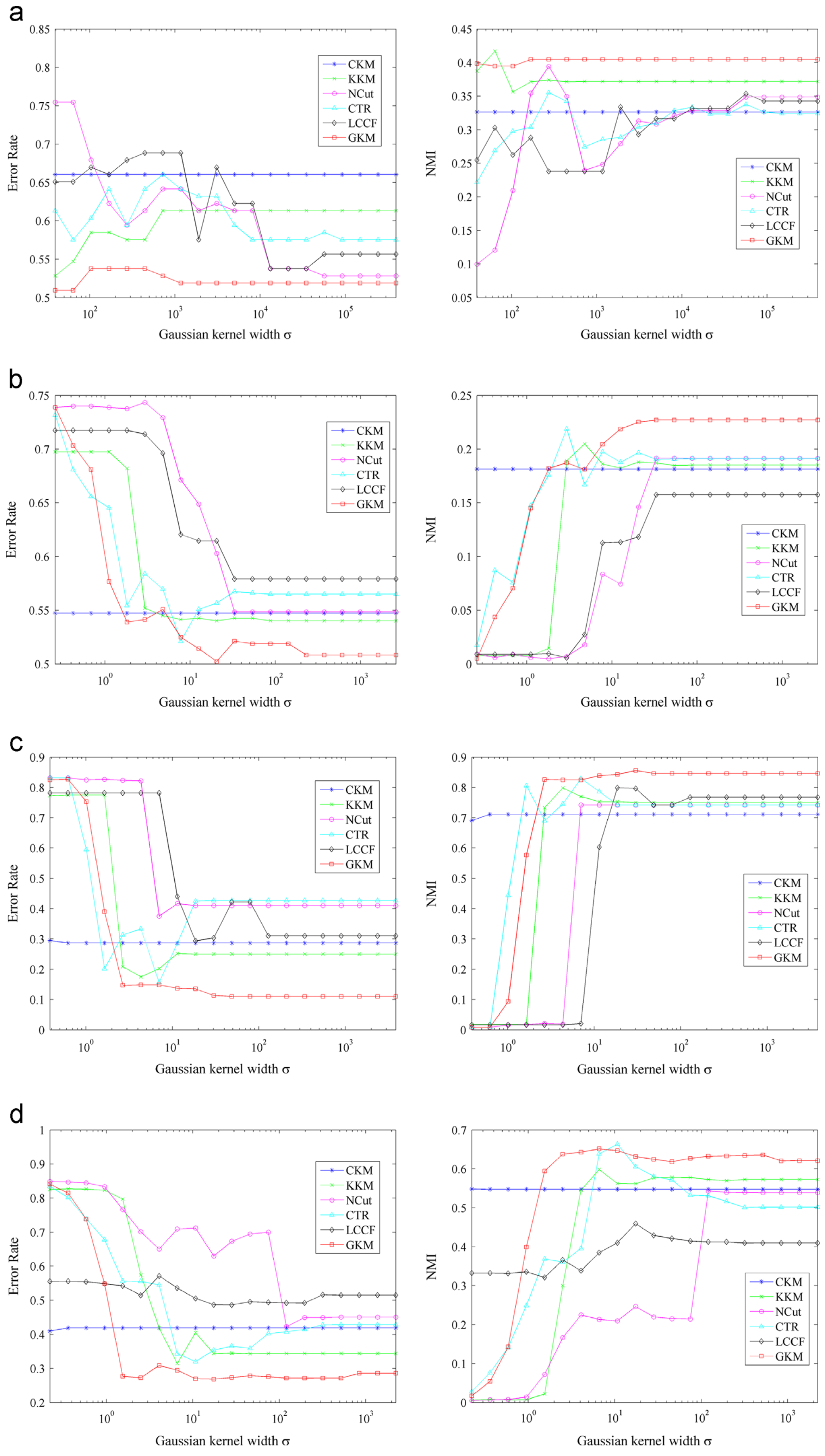


Fig. 9. Experimental results of four real world data sets from UCI repository. (a) breasttissue; (b) vehicle; (c) control data; and (d) statlog.

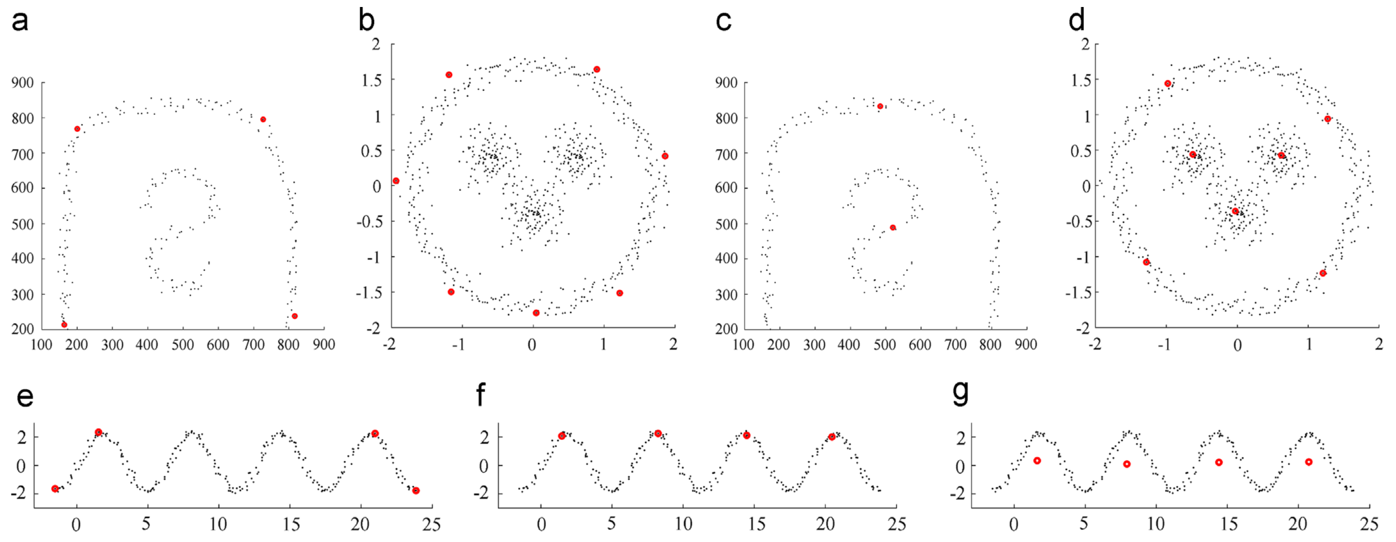


Fig. 10. Results of representative selection on toy data sets (red circles are representatives). (a) (b) (e) are the results of SMRS; (c) (d) (f) are the results of GKM; (g) is the result of CKM. Note the difference between representatives defined, thus found, by SMRS and GKM: for SMRS, the representatives are the vertices of the convex hull of the data set, ignoring the inner structure of the data sets; for GKM, the representatives are the points that minimize the total sum of geodesic distances between representatives and their supporters. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

The results show (1) by taking the manifold structure into account, GKM can greatly improve the performance of the classic *k*-means; and (2) the error curve of GKM decreases earlier and faster than that of the baseline algorithms, indicating that GKM can endure a wider range of parameter changes.

5.5. *Results of representative selection*

We compare the results given by GKM with the results of the very recently proposed Sparse Model Representation Selection algorithm (SMRS)[15] [4] to demonstrate the power of GKM in selecting representative samples. Fig. 10 shows the experimental results of the synthetic data sets and Fig. 11 shows the results of real-world data sets. The results show that GKM differs from SMRS in the following aspects: (1) SMRS finds the vertices of the convex hull of the data cloud but pays little attention to the clouds inner structure, while GKM extracts representatives with respect to the density of regions in the clouds, i.e. the representatives are those that have compact and dense supporters around. SMRS is advantageous while the data set poses a solid convex shape. But manifold-distributed data sets present in various forms and inner structures, which convey important information about data characteristics. (2) SMRS automatically determines the number of representatives, while GKM can be set flexibly to any number less than the data set size. Often, one may prefer to manually set the number of representatives. For example, in a public contest, one may want to choose the top three of all contestants that can best meet the assessment criteria. (3) SMRS solves an unsmooth optimization problem and thus scales badly for large data set, while GKM costs little in iterations, given the distance matrix and accumulated transition probability matrix. As a comparison, we also apply classic *k*-means CKM to the third data set and the result is shown in Fig. 10(g). As previously mentioned, the centroids given by CKM are generally not on the manifold.

5.6. *Time complexity comparison*

We carry out experiments on USPS data set[16] to examine the time complexity of GKM implemented by straightforward method (i.e., using full matrix inversion and Floyd's algorithm to compute $\overline{P}$ and D_G, respectively) and efficient method (i.e., using Nyström approximation and Dijkstra algorithm to exploit sparsity of k NN graph to compute $\overline{P}$ and D_G, respectively). The USPS data set contains 9298 images of handwritten digits from 0 to 9. All images are normalized to 16×16. The feature vector of each image is formed by concatenating all the columns of the image intensity. We compare the time and error rate of the straightforward and the efficient implementations. We randomly choose subsets with different sizes from the USPS data set and run straightforward and efficient GKM on these subsets, fixing the kernel width parameter σ to $9.5r$. For each subset, the parameter N is set to 10 in constructing k NN graph and the parameter m in Nyström method is set to 5% of the subset size. Experimental results are shown in Fig. 12. From these results we can see that the efficient implementation scales well in terms of data set size. It can achieve a comparable performance with the straightforward implementation while its computational complexity is much lower.

We also conduct experiments on the *5% Training* subset of 20 newsgroups database to compare the time complexity of GKM and other baseline algorithms. All the algorithms are run on a computer with 4 GB RAM and 2.67 GHZ CPU. Time cost is shown in Table 2. While the straightforward GKM has a relative high time cost, the efficient GKM has the lowest time cost among all the algorithms.

5.7. *Effects of graph construction and distance calculation*

In this section we conduct three experiments to investigate the effects of graph construction and distance calculation upon GKM clustering performance. There are two parameters of graph construction, the edge weight kernel width σ and the neighborhood size N for k NN graph construction. For graph distance calculation, there is also a neighborhood size parameter for constructing the Euclidean distance neighborhood graph. In order to distinguish one from another, we denote N_W the neighborhood size for graph construction and N_D the neighborhood size for graph distance matrix computation. In the first experiment, we change N_W from 2 to 20 and set $\sigma{=}0.5r$, $N_D{=}6$ to observe the trend of clustering performance. In the second experiment, contrarily we let N_D change from 2 to 20 and set $\sigma = 0.5r$, $N_W{=}6$. In the third experiment we set both N_W and N_D to 6, and change σ from $0.01r$ to $100r$. Experimental results are shown in Fig. 13.

[15] Software:http://www.cis.jhu.edu/ehsan/code.htm.
[16] http://www-stat.stanford.edu/~tibs/ElemStatLearn/.

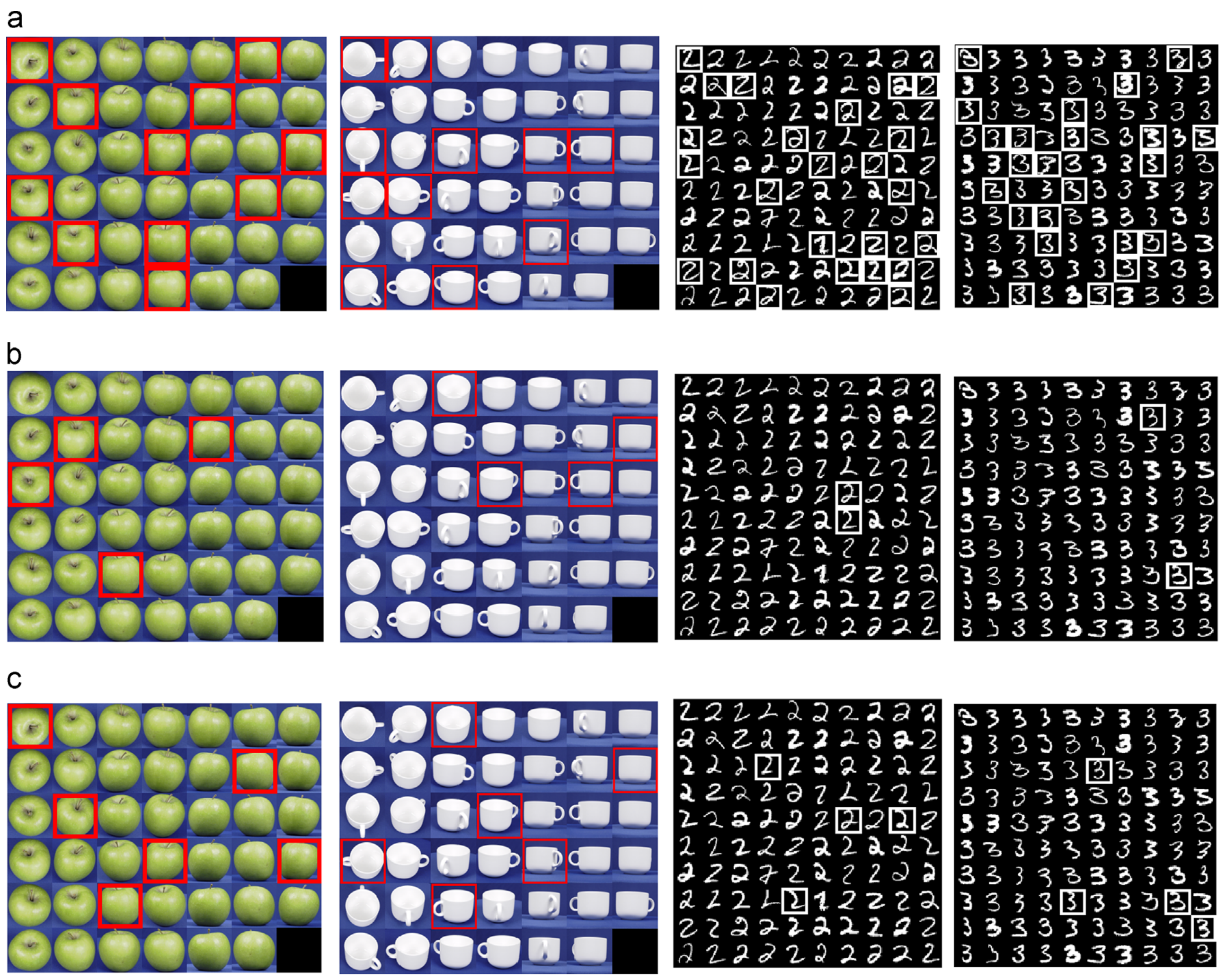


Fig. 11. Results of representative selection on real-world data sets. (a) The results of SMRS; (b) and (c) The results of GKM for different numbers of representatives. Left-hand data sets: apple1 and cup6 in ETH-80 database; right-hand data sets: the first 100 samples of digits 2 and 3 from MNIST data set.

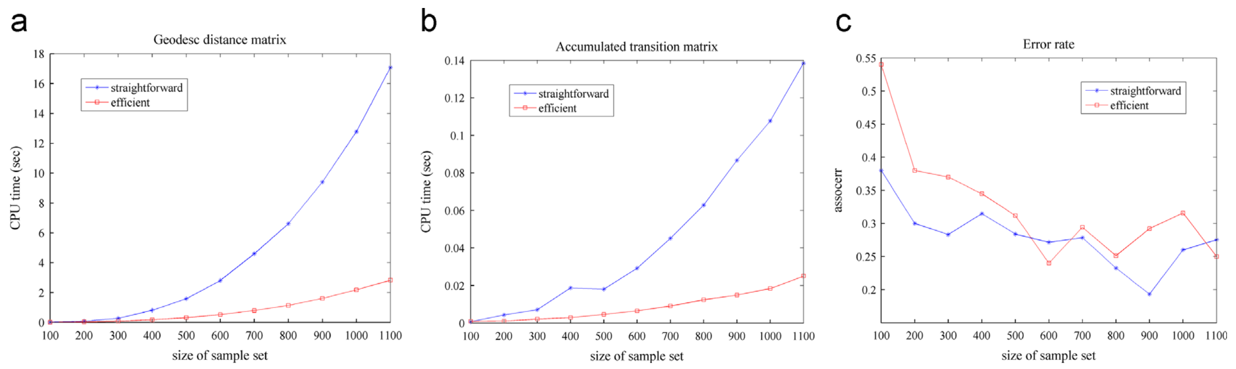


Fig. 12. Results of time complexity comparison experiment. (a) CPU time for computing geodesic distance matrix; (b) CPU time for computing accumulated transition probability matrix; and (c) error rate of straightforward and efficient ways.

From these results we can make the following conclusions: (1) GKM clustering performance is very stable to N_W and N_D. Because the tired random walk model takes all the possible pathes between any two vertices into consideration, it is much stable to a local change in the graph. So the clustering performance is also stable to the local change. (2) The kernel width has a big effect on clustering performance while it is small. But as the kernel width gets large, the error rate decreases rapidly and becomes stable. The reason is that in a k NN graph, the edge weight will be very weak (almost 0 because of the rapid decay of exponential function) while the kernel width is very small. These weak connections usually are not enough to reflect the tightness between the neighboring samples and the manifold structure information,

Table 2
Time cost of the algorithms (s). S, straightforward; E, efficient.

Algorithm	CKM	KKM	NCut	CTR	LCCF	GKM(S)	GKM(E)
Time cost	14.81	16.30	26.09	70.84	49.69	61.58	11.05

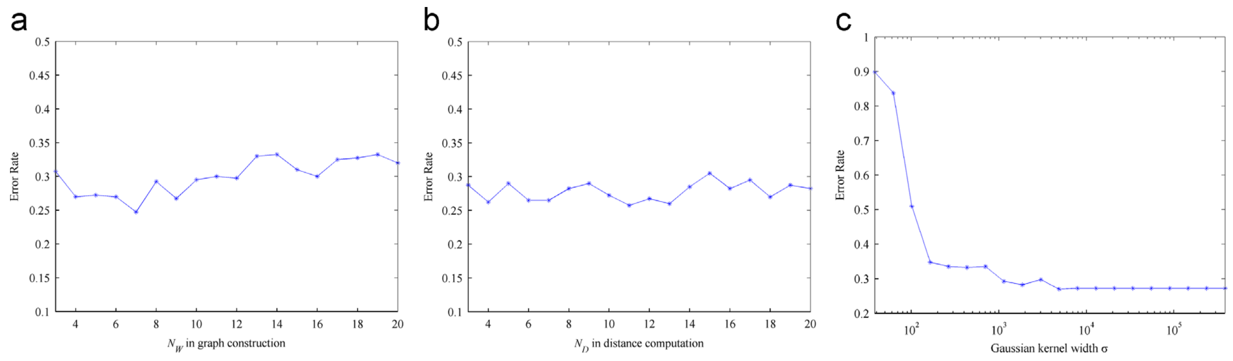


Fig. 13. Effect of graph construction and distance calculation upon clustering performance. (a) Error rate vs. N_W; (b) Error rate vs. N_D; and (c) Error rate vs. σ.

and thus the performance can be very poor. While the kernel width gets large, the connection weight of the graph will increases quickly. Neighboring samples are thus connected more tightly than far away samples and the performance gets better. As the kernel width increase continuously, the connections in the graph become strong and strong. The limit case is that all the connection weights between neighboring samples are finally become 1 as the kernel width approaches to infinity. In this case the weighted k NN graph becomes a regular unweighted k NN graph. Nevertheless, this unweighted k NN graph still encodes the manifold information [31,30] and guarantees the performance to be stable.

6. Conclusions and future work

In order to overcome the drawbacks of classic k-means in dealing with manifold-distributed data structure, we have demonstrated a graph-based k-means (GKM) which can effectively perform manifold clustering and representative selection while remaining a similarly simple iterative process as the classic k-means. Substantial experiments have been carried out on both synthetic and real-world data sets with very promising results. GKM has the advantage of the popularity of the k-means type of clustering and also effectively caters for the manifold-distributed data structure widely existing in practice. This makes GKM very promising for handling real-life applications. However, similar to the classic k-means and kernel k-means, GKM also suffers from the initial conditions which have a significant influence on the final results. Although the initialization method in Section 5.3 can improve the performance stability, this is still an open problem that is worth further study. We are now working on theoretical analysis of the proposed tired random walk model and designing more appropriate settings for initial conditions to reduce the impact and extending GKM to a pair-wise constrained learning framework. Our future work will also cover how GKM could effectively cluster overlapping manifolds, since clustering overlapping manifold is one of the most challenging problems for many manifold clustering algorithms.

Acknowledgments

This research is partly supported by NSFC China (No. 61273258) and Ph.D. Programs Foundation of Ministry of Education of China (No. 20120073110018).

References

[1] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, California, USA, 1967, pp. 281–297.
[2] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[3] J.B. Tenenbaum, V. De Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[4] E. Elhamifar, G. Sapiro, R. Vidal, See all by looking at a few: sparse modeling for finding representative objects, in: CVPR, IEEE, 2012, pp. 1600–1607.
[5] E. Elhamifar, G. Sapiro, R. Vidal, Finding exemplars from pairwise dissimilarities via simultaneous sparse recovery, in: NIPS, vol. 2, 2012, pp. 1–9.
[6] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, Cambridge, UK, 2004.
[7] D. Zhou, J. Weston, A. Gretton, O. Bousquet, B. Schölkopf, Ranking on data manifolds., in: NIPS, vol. 3, 2003.
[8] R. Subbarao, P. Meer, Nonlinear mean shift for clustering over analytic manifolds, in: CVPR, vol. 1, IEEE, 2006, pp. 1168-1175.
[9] R. Subbarao, P. Meer, Nonlinear mean shift over Riemannian manifolds, Int. J. Comput. Vis. 84 (1) (2009) 1–20.
[10] A. Goh, R. Vidal, Segmenting motions of different types by unsupervised manifold clustering, in: CVPR, IEEE, 2007, pp. 1–6.
[11] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[12] A.Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering1 analysis and an algorithm, NIPS 14 (2001) 849–856.
[13] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering., in: NIPS, vol. 14, 2001, pp. 585–591.
[14] P. Li, J. Bu, B. Xu, B. Wang, C. Chen, Locally discriminative spectral clustering with composite manifold, Neurocomputing 119 (2013) 243–252.
[15] S. Wu, X. Feng, W. Zhou, Spectral clustering of high-dimensional data exploiting sparse representation vectors, Neurocomputing 135 (2014) 229–239.
[16] M. Breitenbach, G. Z. Grudic, Clustering through ranking on manifolds, in: ICML, ACM, 2005, pp. 73–80.
[17] R. Souvenir, R. Pless, Manifold clustering, in: ICCV, vol. 1, IEEE, 2005, pp. 648–653.
[18] E. Elhamifar, R. Vidal, Sparse manifold clustering and embedding, in: NIPS, 2011, pp. 55–63.
[19] S. Yin, S.X. Ding, A. Haghani, H. Hao, P. Zhang, A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark tennessee eastman process, J. Process Control 22 (9) (2012) 1567–1581.
[20] S. Yin, H. Luo, S.X. Ding, Real-time implementation of fault-tolerant control systems with performance optimization, IEEE Transactions on Industrial Electronics 61 (5) (2014) 2402–2411.
[21] R. Maitra, I.P. Ramler, A k-mean-directions algorithm for fast clustering of data on the sphere, J. Comput. Graph. Stat. 19 (2) (2010) 377–396.
[22] G.D. Canas, T. Poggio, L. Rosasco, Learning manifolds with k-means and k-flats, in: NIPS, 2012, pp. 2474–2482.
[23] A. Schenker, Graph-Theoretic Techniques for Web Content Mining, vol. 62, World Scientific, 5 Toh Tuck Link, Singapore, 2005.
[24] L. Galluccio, O. Michel, P. Comon, A.O. Hero III, Graph based k-means clustering, Signal Process. 92 (9) (2012) 1970–1984.
[25] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, in: NIPS, vol. 16, 2003, pp. 321–328.
[26] X. Zhu, Z. Ghahramani, J. Lafferty, et al., Semi-supervised learning using gaussian fields and harmonic functions, in: ICML, vol. 3, 2003, pp. 912–919.
[27] M.L. Fredman, R.E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, J. ACM (JACM) 34 (3) (1987) 596–615.
[28] C. Williams, M. Seeger, Using the Nyström method to speed up kernel machines, in: NIPS, 2001.
[29] D. Cai, X. He, J. Han, Locally consistent concept factorization for document clustering, IEEE Trans. Knowl. Data Eng. 23 (6) (2011) 902–913.
[30] X. Zhu, A.B. Goldberg, Introduction to semi-supervised learning, Synthesis Lectures on Artificial Intelligence and Machine Learning 3 (1) (2009) 1–130.
[31] X. Zhu, J. Lafferty, R. Rosenfeld, Semi-Supervised Learning with Graphs (Ph.D. Thesis), Carnegie Mellon University, Language Technologies Institute, School of Computer Science, 2005.
[32] S.X. Yu, J. Shi, Multiclass spectral clustering, in: ICCV, IEEE, 2003, pp. 313–319.
[33] F. Wang, C. Zhang, Label propagation through linear neighborhoods, IEEE Trans. Knowl. Data Eng. 20 (1) (2008) 55–67.
[34] H. Valizadegan, R. Jin, Generalized maximum margin clustering and unsupervised kernel learning, Adv. Neural Inf. Process. Syst. 19 (2007) 1417.

Enmei Tu was born in Anhui, China. He received his B. Sc. degree and M.Sc. degree from University of Electronic Science and Technology of China (UESTC) in 2007 and 2010, respectively. He is now a PhD candidate in the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, China. His research interests are machine learning, computer vision and neural information processing.

Longbing Cao got one Ph.D. in Intelligent Sciences in Chinese Academy of Sciences and another in Computing Science from the University of Technology Sydney (UTS). He is a professor of information technology at UTS, the Founding Director of the University's Research Institute Advanced Analytics Institute and a core member of the Data Sciences and Knowledge Discovery Lab at the Centre for Quantum Computation and Intelligent Systems at the Faculty of Engineering and IT, UTS. He is also the Research Leader of the Data Mining Program at the Australian Capital Markets Cooperative Research Centre, the Chair of IEEE Task Force on Behavior and Social Informatics and of IEEE Task Force on Educational Data Mining. He is a Senior Member of IEEE, SMC Society and Computer Society.

Jie Yang was born in Shanghai, China, in August 1964. He received a bachelor's degree and a master's degree in Automatic Control in Shanghai Jiao Tong University in 1985 and 1988, respectively. In 1994, he received Ph. D. at Department of Computer Science, University of Hamburg, Germany. Now he is the Professor and Director of Institute of Image Processing and Pattern Recognition in Shanghai Jiao Tong University. He is the principal investigator of more than 30 nation and ministry scientific research projects in image processing, pattern recognition, data mining, and artificial intelligence, including two national 973 research plan projects, three national 863 research plan projects, three national nature fundation projects, five international cooperative projects with France, Korea, Japan, New Zealand. He has published more than 500 of articles in national or international academic journals and conferences.

Nikola Kasabov is a Fellow of the Royal Society of New Zealand, the New Zealand Computer Society and the Institute of Electrical and Electronic Engineers (IEEE). He is the founding Director and the Chief Scientist of the Knowledge Engineering and Discovery Research Centre (KEDRI) and Personal Chair of Knowledge Engineering in the School of Computing and Mathematical Sciences at AUT. His main interests are in the areas of computational intelligence, neuro-computing, bioinformatics, neuroinformatics, speech and image processing, novel methods for data mining and knowledge discovery. He has published over 450 works, among them journal papers, text books, edited research books and monographs, conference papers, book chapters, edited conference proceedings, patents and authorship certificates in the area of intelligent systems, connectionist and hybrid connectionist systems, fuzzy systems, expert systems, speech recognition, bioinformatics, neurocomputing and neural networks.