Contents lists available at ScienceDirect

Neurocomputing

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

Quantum clustering using kernel entropy component analysis

Yangyang Li [a,*], Yang Wang [a], Yuying Wang [b], Licheng Jiao [a], Yang Liu [c]

[a] Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China, International Research Center for Intelligent Perception and Computation, Xidian University, Xi'an 710071, PR China
[b] IBM (China) Investment Company Limited, Xi'an Branch, Global Business Solution Center, Xi'an 710000, PR China
[c] Department of Information Engineering, North China University of Water Resources and Electric Power, Zhengzhou 450045, PR China

ARTICLE INFO

Article history:
Received 26 June 2014
Received in revised form
27 October 2015
Accepted 15 March 2016
Communicated by E.W. Lang
Available online 13 April 2016

Keywords:
Quantum clustering
Kernel entropy component analysis
K-nearest neighbors

ABSTRACT

In this paper, a novel method quantum clustering using kernel entropy component analysis (KECA-QC) is proposed. This method has two phases: preprocessing and clustering stages. The main idea of preprocessing is to map the original data to a high-dimensional feature space, and to select the useful components using Renyi entropy as our similarity metric. After data preprocessing, different clusters will be distributed more or less in different places, and for high-dimensional datasets, it can achieve the purpose of dimensionality reduction at the same time. In the second phase, quantum clustering method is used, which can find clusters of any shape without knowing the number of clusters. Based on the traditional quantum clustering, we develop a new method estimating the wave function from distributions of *K*-nearest neighbors statistics, which can further reduce the running time and improve the calculation efficiency. In order to evaluate the effectiveness of this method, we compare the proposed method with *k*-means clustering (KM), the classical spectral clustering algorithm called Ng–Jordan–Weiss (NJW), the traditional QC, and kernel entropy component analysis spectral clustering algorithm (KECA-KM). The experimental results demonstrate that the proposed algorithm outperforms the compared algorithms on synthesized datasets and UCI datasets.

1. Introduction

Clustering analysis refers to the process of organizing data into meaningful homogeneous groups or clusters. Its aim is to get high similarity among the objects in the same cluster, but dissimilarity in different clusters. Clustering usually does not need a training data for learning, which is in an unsupervised classification. It has been widely used in pattern recognition, data mining and machine learning. Traditionally, the existing clustering methods can be divided into partitional, hierarchical, density-based, and model-based algorithms. Each algorithm has its own advantages and disadvantages. For example, *k*-means algorithm [1], one of the most famous clustering algorithms, can be implemented easily and efficiently, and have been successfully employed in many practical applications. However, this algorithm only works well for hyper-spherical data, or at best hyper-elliptical data. In the existing clustering algorithms, including *k*-means algorithm, most partitional approaches require a-priori knowledge about the number of clusters the data should be grouped into. However, in many cases, the number of clusters is unknowable. Moreover, the vast majority of clustering methods have many drawbacks, such as sensitivity to their initialization and noise and susceptivity to local optima.

Inspired by quantum mechanism in physics, we can solve those problems mentioned above using quantum theory. Quantum clustering (QC) [2–4], proposed by David Horn and Assaf Gottlieb, takes the clustering as a physical system. By solving the Schrödinger equation and using the gradient descent method, the resulting potential function has the minimums which correspond to the cluster centers. If the parameters of QC are fixed, it is a deterministic algorithm and belongs to one of the unsupervised clustering algorithms based on partition. Many experiments show that QC can produce satisfactory results even when some traditional clustering algorithms fail.

Nasios et al. concluded that the potential field is assimilated with the data density [5,6]. So they used the *K*-nearest neighbors statistical distribution for estimating the scale parameter, and detected the final modes by combining the local Hessian with a region growing algorithm. Li et al. improved the QC algorithm and proposed the distance-based QC [7] and parameter-estimated QC [8], to overcome the defect of the traditional QC, that is, the measured distance between two samples is relatively fixed and the kernel scale parameter is often needed to be estimated by experiments many times. Subsequently, Zhang et al. substituted the exponent measuring distance for Euclidean distance to

* Corresponding author. Tel.: +86 02988202279.
E-mail address: yyli@xidian.edu.cn (Y. Li).

http://dx.doi.org/10.1016/j.neucom.2016.03.006
0925-2312/© 2016 Elsevier B.V. All rights reserved.

measure the distance between data points and the cluster centers [9], which improved the iterative procedure of QC algorithm and performed better than the Euclidean distance in data preprocessing. Gou et al. made a combination of quantum clustering and multi-elitist immune algorithm [10,11] to avoid getting stuck in local extremes the and solve the computational bottleneck. In addition, quantum clustering combination with other methods can be applied to many different areas. In Refs. [8] and [11], the improved quantum clustering algorithms are applied for the topography segmentation from SAR images of terrain and medical images segmentation respectively. Niu et al. detected the community structure in complex networks based on quantum mechanics [12]. Sun et al. applied quantum clustering to the research of fuzzy neural network model [13]. Di Buccio et al. distilled relevant documents by means of dynamic quantum clustering [14].

In this paper, we introduce a new quantum clustering using kernel entropy component analysis [15,16]. Firstly, we use kernel entropy component analysis as a data preprocessing stage, which is to map the original data to a high-dimensional feature space and to select the useful components substituting original data. It is proved that all the information contained in the distribution of the data can be utilized by using entropy as our metric. Our approach is capable of finding clusters of any shape, without knowing the real number of clusters beforehand. The clustering results of datasets especially high-dimensional data are obviously improved after data preprocessing. In addition, quantum clustering may take a long period of time to execute. So in order to reduce the running time and improve the calculation efficiency, the distributions of K-nearest neighbors statistics are used for calculating the wave function of each quantum physics particle. Meanwhile, the introduction of local information can help improve the quantum clustering accuracy.

The rest part of this paper is organized as follows. In the next section, the quantum clustering algorithm is described. Section 3 gives the details of the proposed data preprocessing, KECA, and then the improved quantum clustering with K-nearest neighbors is described. Section 4 provides the experimental results and discussions about our algorithm. The conclusions of this study are drawn in the last part of this paper.

2. Related work – quantum clustering algorithm

Quantum mechanics describes the distribution of particles in quantum space, and clustering analysis detects the structure of samples in scale space. In this case, we can observe the similarity between data points and quantum particles. The study of quantum physicists found that the distribution of microscopic particles in the energy field is influenced by their potential energy. When the spatial distribution and evolution shrink down to a one-dimensional infinite square potential well, the particles always tend to gather together at low potential values [17]. In other words, potential function is equivalent to an abstract source, and there are more particles distributed in the potential well when the data potential function reaches minimum. Thus, quantum mechanics with this process can be used to solve clustering problems. Particles in the same cluster will eventually be placed in the same potential well but different wells for particles belonging to different cluster. There is corresponding relation between particles and data points, and we can convert the data clustering problem into a quantum clustering issues.

Quantum clustering algorithm [2–4] proposed by Horn is a novel clustering method that is based on physical intuition derived from quantum mechanics. The state of a quantum mechanics system is completely specified by the wave function [6]. According to the theory of quantum mechanics, the evolution of quantum states follows the Schrödinger equation. The Stationary Schrödinger equation can be represented as

$$H\psi = \left(-\frac{\sigma^2}{2}\nabla^2 + V(\boldsymbol{p})\right)\psi = E\psi \tag{1}$$

where $\psi(\boldsymbol{p})$ is the wave function, $V(\boldsymbol{p})$ is the potential function, H is the Hamilton operator, E is the energy eigenvalue of H, ∇ is the Laplacian operator, and σ represents the adjustable scale parameter.

From above we can conclude that particles have the same distribution state if they are at the same potential. If the wave function $\psi(\boldsymbol{p})$ is given, the potential function $V(\boldsymbol{p})$ can be worked out with the Schrödinger equation. The minima of $V(\boldsymbol{p})$ can be associated with cluster centers. The process above provides the basis for quantum clustering.

Such a wave function $\psi(\boldsymbol{p})$ can be estimated with the Gaussian kernel-based sum.

$$\psi(\boldsymbol{p}) = \sum_{i=1}^{N} e^{-\|\boldsymbol{p}-\boldsymbol{p}_i\|^2/2\sigma^2} \tag{2}$$

In total, there are N quantum particles in quantum space and $\boldsymbol{p}_i$ is one of them. Working out the Eq. (1) with the given $\psi(\boldsymbol{p})$, we can get the general expression of potential function as

$$V(\boldsymbol{p}) = E + \frac{(\sigma^2/2)\nabla^2\psi}{\psi} = E - \frac{d}{2} + \frac{1}{2\sigma^2\psi}\sum_i \|\boldsymbol{p}-\boldsymbol{p}_i\|^2 \exp\left[-\frac{\|\boldsymbol{p}-\boldsymbol{p}_i\|^2}{2\sigma^2}\right] \tag{3}$$

where d is the lowest possible eigenvalue of H, which is the dimension of data points [18]. For E is still left undefined, we require V to be nonnegative, i.e. min $V = 0$. E can be approximated by

$$E = -\min \frac{(\sigma^2/2)\nabla^2\psi}{\psi} \tag{4}$$

Refs. [2,3] use the gradient descent algorithm to find the quantum potential minima as the cluster centers. The iterative formula is as following:

$$\boldsymbol{y}_i(t+\Delta t) = \boldsymbol{y}_i(t) - \eta(t)\nabla V(\boldsymbol{y}_i(t)) \tag{5}$$

where $\boldsymbol{y}_i(0) = \boldsymbol{p}_i$ is the initial data, $\eta(t)$ is the iteration speed and ∇V is the gradient of potential function. Eventually, data point $\boldsymbol{y}_i$ reaches a fixed value which is consistent to the cluster center. And some nearest data points should be classified into the same class.

It follows therefore that, the potential function is the cost function in QC. The location of the cluster centers in the clustering process depends on the potential structures of the data, but need not define the geometric centers or random data points as our cluster centers.

3. Quantum clustering using kernel entropy component analysis (KECA-QC)

3.1. Kernel entropy component analysis

In this subsection, the kernel entropy component analysis will be introduced. Then the performance of kernel principal component analysis (KPCA) [19,20] and singular value decomposition (SVD) [2,3], which are also widely used preprocessing methods, will be shown. And the comparison and discussion will be made with these three methods.

In 2006, from the view of information theory, Jenssen et al. combined Renyi entropy [15] with the kernel method and proposed a novel method called kernel entropy component analysis (KECA). It is able to retain the main information of data structure and has a strong capability of nonlinear data-processing. This feature extraction method is also applied as a preprocessing stage to quantum clustering in our method.

Given a dataset $\boldsymbol{X}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_N\}$, $\boldsymbol{x}_i,\boldsymbol{x}_j\in\boldsymbol{R}^d$, $i,j=1,2,\cdots,N$. We also choose the Gaussian kernel matrix

$$G(\boldsymbol{x}_i,\boldsymbol{x}_j)=e^{-\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2/2\sigma^2} \tag{6}$$

where σ is the only parameter of Gaussian kernel. G is an $(N\times N)$ matrix whose dimension equals to the number of samples in the dataset. Moreover, the eigenvalues and eigenvectors of the corresponding kernel matrix can be eigendecomposed as $G=Q\Lambda Q^T$, where Λ is a diagonal matrix storing the eigenvalues $\lambda_1,\lambda_2,\cdots,\lambda_N$ $(\lambda_1\ge\lambda_2\ge\cdots\ge\lambda_N)$ and Q is a matrix with the corresponding eigenvectors $\boldsymbol{q}_1,\boldsymbol{q}_2,\cdots,\boldsymbol{q}_N$ as columns.

As a similarity metric, the Renyi quadratic entropy helps select which eigenvectors we want. The Renyi entropy is defined as $-\log\int p^2(\boldsymbol{x})d\boldsymbol{x}$, where $p(\boldsymbol{x})$ is the probability density function. Since the logarithm is a monotonic function, we may concentrate on the quantity $\boldsymbol{r}=\int p^2(\boldsymbol{x})d\boldsymbol{x}$. In order to estimate $\boldsymbol{r}$, a Parzen window density estimator is used to get $\hat{p}(\boldsymbol{x})=\frac{1}{N}\sum_i K(\boldsymbol{x},\boldsymbol{x}_i)$ [35]. The value of the quantity $\boldsymbol{r}$ can be evaluated as follows [16]:

$$r=\frac{1}{N}\sum\hat{p}(x)$$

$$=\frac{1}{N^2}\mathbf{1}^T\mathbf{K}\mathbf{1} \tag{7}$$

here $\mathbf{K}=\mathbf{G}$,

$$r=\frac{1}{N^2}\sum_i\left(\sqrt{\lambda_i}\mathbf{q}_i^T\mathbf{1}\right)^2 \tag{8}$$

then the entropy of each eigenvector can be presented as follow:

$$r_i=\left(\sqrt{\lambda_i}\mathbf{q}_i^T\mathbf{1}\right)^2 \tag{9}$$

where $\mathbf{1}$ is an $(N\times1)$ vector and each element equals one. Eq. (9) shows that the entropies corresponding to certain eigenvalues and eigenvectors will be greater than others since the entropies depend on different eigenvalues and eigenvectors.

Then, the eigenvectors and corresponding eigenvalues are ranged in decreasing order of the entropies. The kernel entropy component analysis (KECA) are defined as an l-dimensional data transformation. Λ_{keca_l} consists of the top l corresponding eigenvalue and Q_{keca_l} stores the corresponding eigenvectors as columns. Therefore, the resulting KECA expression becomes

$$\Phi_{\text{keca}}=\Lambda_{\text{keca}\_l}^{\frac{1}{2}}Q_{\text{keca}\_l}^T \tag{10}$$

Φ_{keca} is the $(N\times l)$ matrix storing the kernel entropy components and it is used as the initial data for quantum clustering. Each row in this matrix represents one point in the original dataset.

The following part of this section helps explain why we choose to use KECA as the preprocessing, rather than other methods like SVD and KPCA. Firstly, we take two synthetic datasets shown in

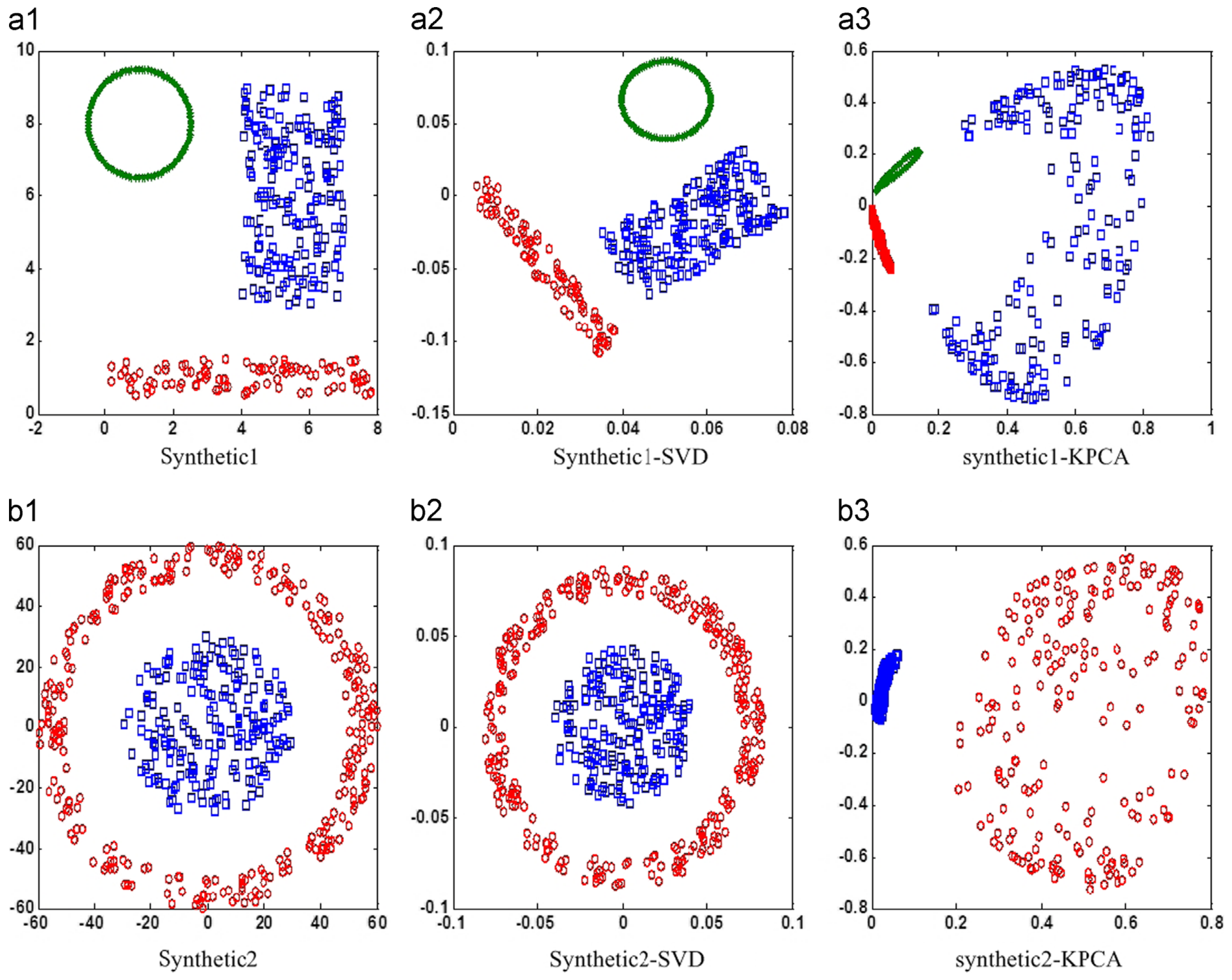

Fig. 1. Data transformations using SVD and KPCA.

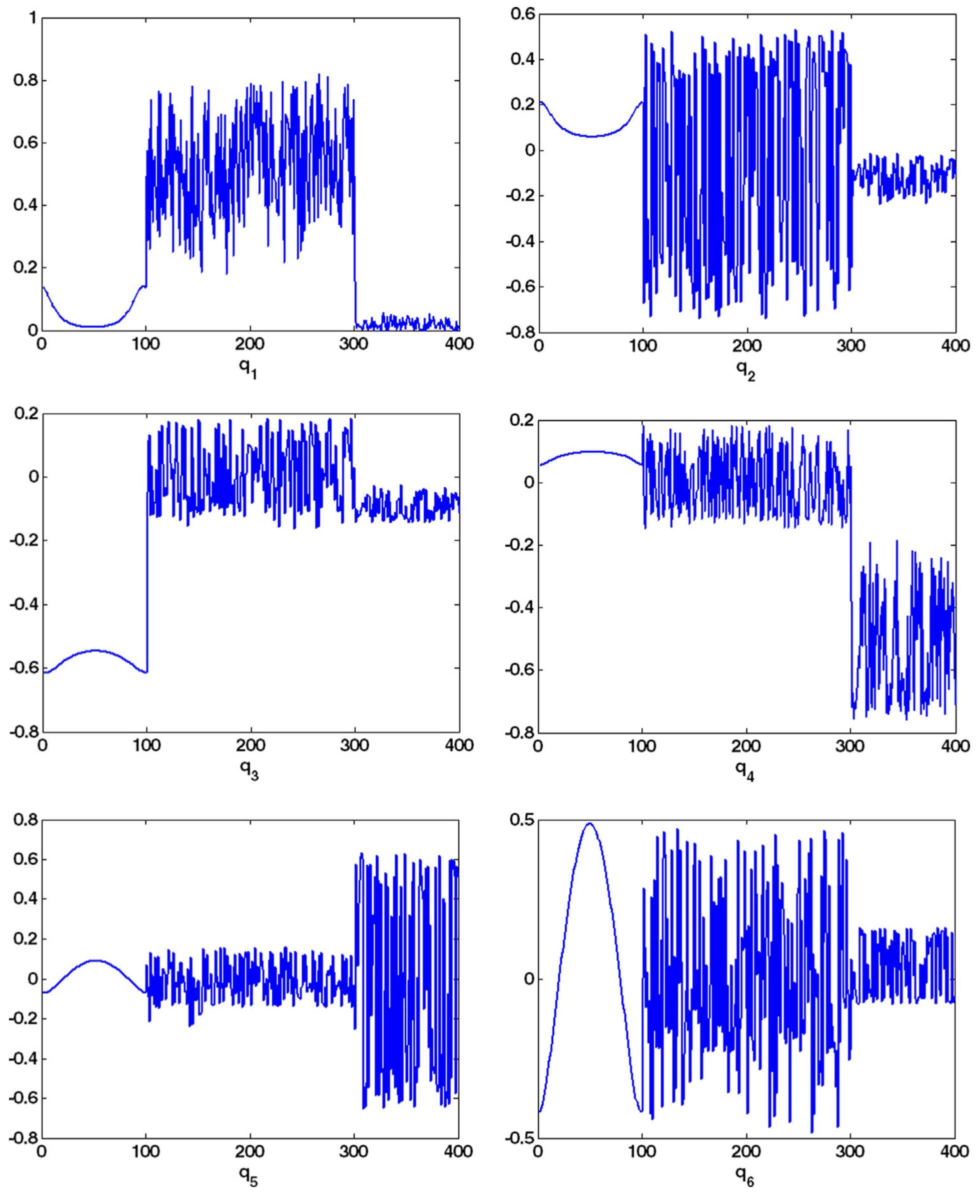

Fig. 2. The projection of samples on the top six eigenvectors in decreasing order of the eigenvalues.

Fig. 1(a-1 and b-1) as examples to show data preprocessing using SVD and KPCA, as the representative of linear and nonlinear transformation, respectively. Among all the panels, (a-2, a-3, b-2 and b-3) represent the data projections of two-dimensional map. Obviously we can see from these panels that the linear transformation and the nonlinear transformation with the introduction of Gaussian kernel are obviously different. SVD cannot separate the different categories, while KPCA is good at seizing the main information of data structure particularly obvious on the relatively complex data (such as manifold data). KPCA completely separates two clusters of Synthetic2, but SVD could not.

In KPCA-preprocessing, we always selects the top l eigenvectors corresponding the l largest eigenvalues of the matrix G. However, it is not guaranteed that the structure of the data can be well detected by the largest l eigenvectors [21,22]. Fig. 2 takes Synthetic1 as an example and shows the projection of samples on the top six eigenvectors ($\boldsymbol{q}_1 \sim \boldsymbol{q}_6$) of the kernel matrix. In all the panels, some elements have non-zero value. The remaining elements are basically zero. If three eigenvectors are used, although the nonzero elements in $\boldsymbol{q}_1$ are corresponding to the same elements in $\boldsymbol{q}_2$, both the eigenvectors will be selected in the traditional methods. But with KECA, the top three eigenvectors are $\boldsymbol{q}_1$, $\boldsymbol{q}_3$ and $\boldsymbol{q}_4$. From Fig. 2, we can see that those three eigenvectors have different nonzero parts. With the nonzero elements of $\boldsymbol{q}_1$, $\boldsymbol{q}_3$ and $\boldsymbol{q}_4$, we can get the information to distinguish those three different clusters.

For a dataset with N samples, an $N \times N$ kernel matrix can be obtained through Gaussian kernel transformation and then its eigen-decomposition is performed. At the same time, Renyi entropy, as a similarity metric, is used to select which eigenvectors we want, and finally, the transformed data consisting of kernel entropy components are created by removing the rest dimensions (redundant information). In general, KECA preprocessing mentioned above, is a process of reducing after raising the dimension of the data. KECA not only helps to get compact and economical representations of data structure, but also provides some other advantages, such as better expression of data characteristics and expanding the differences between different data clusters. And KECA has the ability of transforming a nonlinear separable problem into a linear separable problem. Those are all conducive to improving the cluster accuracy. It is worth noting that KECA achieves data dimensionality reduction for those high-dimensional datasets. But it does not mean that for all datasets, the transformed data produced by KECA has a lower dimension comparing with the input data, like some low-dimensional datasets. Although performing the preprocessing still take some time, a expression form beneficial for clustering is obtained, which can reduce the run-time of the subsequent clustering algorithm and improve the calculation efficiency and cluster accuracy.

3.2. Quantum clustering with K-nearest neighbors

In QC algorithm, Φ_{keca} can be viewed as an input matrix with each row as a particle representing a data point in the original dataset. As can be seen from Eq. (2), the wave function is estimated by summing up all the values of Gaussian kernel function with the distance to other data points. In other words, the wave function of a particle is cooperatively affected by other particles. The farther apart the points in the Euclidean space, the less the effect on their wave function. There is little or no effect on the wave function when the distance is greater than a certain point. Assuming that there are N data points in the dataset, in each iteration, N wave functions should be calculated and each wave function is estimated with all of the N data points. So, the time complexity of the wave function operation is $O\left(N^2\right)$. With the increasing amount of data, the execution time of each iteration is growing exponentially.

In this paper, a new statistical approach for calculating the wave function is introduced. As well known, the values of standard Gaussian kernel function are much greater than 0 but declining rapidly in the interval [0, 3]. And the values are almost 0 when the distance is greater than a certain point near point 3. It means that little or no effect stacked on the wave function from a further distance. Therefore, Gaussian kernel function is often applied to low pass filters. A straight forward consideration is to assume that the wave function of one data point is only affected by the neighbor points. Under this assumption, the wave function can be estimated from the statistics of Gaussian kernel distances between samples from localized data sets. The K-nearest neighbors strategy has been successfully used in pattern recognition for taking into account the local structure of data set in the estimation process [26]. In our approach, we calculate the wave function by using the K-nearest neighbors. Considering a dataset with N data points, $\boldsymbol{p}$ is a random sample from it, and the K-nearest neighborhood of $\boldsymbol{p}$ is $\Gamma_K(\boldsymbol{p})$. We rank all the other data points according to their Euclidean distance to $\boldsymbol{p}$:

$$\left\{\boldsymbol{p}_{(k)} \mid \|\boldsymbol{p}_{(k)}-\boldsymbol{p}\|^2 \le \|\boldsymbol{p}_{(k+1)}-\boldsymbol{p}\|^2, k=1,2,\cdots,K\right\} \tag{11}$$

where $K \le N$ and $\Gamma_K(\boldsymbol{p})=\left\{\boldsymbol{p}_{(1)},\boldsymbol{p}_{(2)},\cdots,\boldsymbol{p}_{(K)}\right\}$ represents the K-nearest neighborhood of $\boldsymbol{p}$. $\boldsymbol{p}_{(1)}$ is the closest data point in the entire data set, while $\boldsymbol{p}_{(K+1)}$ is outside the chosen K-nearest neighborhood data set. We calculate the wave function for the point of $\boldsymbol{p}_i$ corresponding to its neighbors

$$\psi(\boldsymbol{p}) = \sum_{\boldsymbol{p}_{(k)} \in \Gamma_K(\boldsymbol{p})} e^{-\|\boldsymbol{p}-\boldsymbol{p}_{(k)}\|^2/2\sigma^2} \tag{12}$$

Eq. (12) is used as a replacement for the original Eq. (2) to recalculate the wave function. In this formula, we only deal with the points in a limited region, including the center point and its neighbors. With the Eq. (12) plugging into the Schrödinger equation, the final expression for potential function is

$$V(\boldsymbol{p}) = E-\frac{d}{2}+\frac{1}{2\sigma^2\psi}\sum_{\boldsymbol{p}_{(k)} \in \Gamma_K(\boldsymbol{p})} \|\boldsymbol{p}-\boldsymbol{p}_{(k)}\|^2 \exp\left[-\frac{\|\boldsymbol{p}-\boldsymbol{p}_{(k)}\|^2}{2\sigma^2}\right]. \tag{13}$$

As can be seen from Eq. (13), the calculating equation of $V(\boldsymbol{p})$ is only dependent on a neighborhood instead of all data. It has much lower computational costs demand compared with Eq. (3). Although the calculation of k-nearest neighbors needs some time, it is certainly worthy of these times when compared with the statistics of all the values of Gaussian kernel distances. At the same time, the introduction of this local information also helps to improve the accuracy of quantum clustering. In Section 4.3.2, we make a comparison between the kernel entropy component quantum clustering with K-nearest neighbors and this algorithm without K-nearest neighbors on the accuracy and run-time of clustering. The results confirm our conclusion.

3.3. Main loop of KECA-QC

In combination with QC in the previous section described, the proposed algorithm, quantum clustering using kernel entropy component analysis, is a two-stage process. Specifically, we first map the original data set to a kernel space and extract useful components using entropy as a similarity metric. The data points can be considered the particles in quantum space. Then, we make a combination with QC algorithm with the introduction of K-nearest neighbors. The clustering results could be obtained by iterations of the gradient descent algorithm. The main procedure of KECA-QC consists of the following steps:

```
KECA-QC algorithm

Preprocessing stage:
Step 1: For a given data set, calculate the kernel matrix $G$ using
    Eq. (6).
Step 2: Calculate the eigenvalues and eigenvectors of $G$, and
    estimate Renyi entropies those eigenvalues and the corresponding eigenvectors contribute to. Reorder those eigenvalues and the corresponding eigenvectors depending on the
    value of entropies from largest to smallest.
Step 3: Determine the value of parameter $l$, and form the
    matrix $\Phi_{\text{keca}} \in \Re^{N\times l}$ storing $l$ kernel entropy components.
    Treat each row of $\Phi_{\text{keca}}$ as a particle for quantum clustering
    corresponding to one point in the original dataset.
Clustering stage:
Step 4: Calculate the wave function $\psi$, the potential function $V$
    and its gradient descent direction $\nabla V$ of every point with its
    $K$-nearest neighbors using Eqs. (12) and (13).
Step 5: Use the gradient descent algorithm to find the quantum
    potential minima. Iterate it until the number of iterations is
    up to the its maximum Iteration limit, and eventually, the
    data points are located near the cluster centers to which they
    belong.
Step 6: Finally, assign each original point to the cluster whose
    center is closest to this point, and obtain the clustering result
    of the original dataset at the same time.
```

In a broad sense, any algorithm that involves eigen-decomposition for clustering can be called spectral clustering (SC) [23,24]. From this point of view, KECA-QC belongs to spectral clustering. That shows that our algorithm also has the advantages of SC, or we can say the performance of our method is well-matched with SC. The main idea of classical spectral clustering algorithm is as follows: for a L clustering problem, form the affinity matrix of a dataset or compute its Laplacian matrix, select the L largest eigenvalues and the corresponding eigenvectors of Laplacian matrix to form a new matrix, treat each row of it as a point of the initial data, and assign them into L clusters via k-means algorithm to obtain the final clustering result. As can be seen from the steps above, the basic framework of KECA-QC is similar to SC. For this reason, we make a comparison between our algorithm and the representative spectral algorithm, Ng–Jordan–Weiss (NJW) [25]. The clustering results are shown in Section 4.

3.4. Parameters setup

In this algorithm, there are several parameters that should be determined before processing using KECA-QC: two kernel scale parameters σ_{keca} in KECA, σ_{qc} in QC stage, the number of kernel entropy components l, the number of iterations in the gradient descent procedure *Steps*, and the neighborhoods of K.

One of the difficulties within this kernel-based approach is how to select the kernel scale parameter. Because the kernel function observed from Eqs. (2), (6) and (12) can be smooth or over-peaked if the parameter σ is either larger or smaller. Thus the number of clusters in the data implicitly depends on the appropriate selection of σ. So many researchers have conducted research to solve this problem. Quite often, σ is initialized to arbitrary values [2,3], or by employing certain test hypotheses. Silverman introduced an estimation method with the dimension and data size as parameters [29]. Varshavsky used the Bayesian information criterion (BIC) to select the scale parameter σ, and assess the quality of clustering [30]. Nasios et al. thought that the kernel scale can be estimated from distributions of K-nearest neighbors statistics [6]. In this paper, we adopt the idea from Ref. [8,31] for kernel scale estimation. Some necessarily bold assumptions are made that σ is associated with the local data spread. In other word, σ can be calculated from the statistics of Euclidean distance between samples from their K-nearest neighborhood. Under this assumption, we design a simple method to estimate kernel scale parameters. The value $\hat{\sigma}$ is obtained directly by averaging the Euclidean distances from samples to their neighbors.

$$\hat{\sigma} = \frac{\sum_{i=1}^{N} \sum_{\boldsymbol{x}(k) \in \Gamma_K(\boldsymbol{x})} \|\boldsymbol{x}(k) - \boldsymbol{x}\|^2}{N \times (K-1)} \tag{14}$$

Using $\hat{\sigma}$ we evaluate the function G defined in Eq. (6), the wave function $\psi(\boldsymbol{p})$ defined in Eq. (12), and the quantum potential $V(\boldsymbol{p})$ according to Eq. (13). Here, In the calculation of G, the points $\boldsymbol{x}$ and $\boldsymbol{x}(k)$ are the data points in original data set. While evaluating $\psi(\boldsymbol{p})$ and $V(\boldsymbol{p})$, $\boldsymbol{x}$ and $\boldsymbol{x}(k)$ correspond to quantum particles. We sample neighborhoods of $K=50$. The parameter $Steps=20$ employed in our algorithm is specified according to the original papers for the best performance. As is proved by experiment that typically 10 or 20 iterations are enough for QC. The number of components l is determined via a difference method between entropies. Firstly range the entropies of eigenvectors in descending order. Thus, a sequence $\{r_1, r_2, \ldots, r_N\}$ is obtained. Then the l should satisfy the following condition:

$$r_l - r_{l+1} = \max(r_i - r_{i+1}) \quad i = 1, 2, \ldots, N-1 \tag{15}$$

4. KECA-QC for data clustering

In order to evaluate the performance of the proposed algorithm, the proposed algorithms are applied to 18 datasets (including 8 synthesized datasets and 10 UCI datasets [27]). We compare the proposed method with k-means clustering (KM) [1], the kernel k-means clustering (KKM) [32], the classical spectral clustering algorithm called Ng–Jordan–Weiss (NJW) [25], the traditional quantum clustering (QC) [2] and kernel entropy component analysis spectral clustering algorithm (KECA-KM) [16]. The following three metrics are used to evaluate the performance of algorithms: Minkowski Score (MS) [28], Jaccard Score (JS) [2] and cluster accuracy (CA).

Here, KM is one of the simplest unsupervised algorithms that solve the well known clustering problem, and this method gets the final clustering result by defining a value function and finding its global optimum solution. KKM is an alternative of KM algorithm with replacing “Distance” by “Kernel”. Based on the description of SC algorithm in Section 3.3, NJW classifies data using the largest L eigenvectors of the normalized Laplacian matrix derived from the dataset for a L clustering problem. The traditional QC is given in detail in Section 2. KECA-KM is the process of extracting the top L kernel entropy components and then using k-means clustering based on cosine angle distance to divide the original data set into L clusters.

Our experimental environment is: Matlab7.4 (R2007a), Intel (R) Core(TM) 2 Duo CPU 2.33 GHz 2G RAM, Window XP Professional.

In the following experiments, since KM, KKM, NJW and KECA-KM are all stochastic optimization algorithms, the comparison is fair if we calculate the average values of KM, KKM, NJW and KECA-KM from 100 independent runs on each dataset. QC and KECA-QC are both deterministic algorithm when the parameters are determined. In addition, the kernel scale parameters of these contrast algorithms are also set according to Eq. (14).

4.1. Datasets and metrics

Ten synthesized datasets are described in Table 1 and shown in Fig. 3. AD_9_2 and AD_20_2 are both sphere data with a compact distribution. data12 and sizes5 are two data with a more dispersive distribution. data8, eyes, lineblobs and spiral are four data with manifold structure. The description of the ten UCI datasets is given in Table 2, including breastcancer(WBC), german, glass, heart, ionosphere, iris, new_thyroid, sonar, vote and zoo.

In this paper, the following three metrics are used as measures of the quality of a solution given the true clustering. Let T to be the “true” solution and S the solution obtained by those algorithms. Denote by n_{11} the total number of pairs of samples that are in the same cluster in both T and S. The number of pairs that are in the same cluster in T denote by n_{10} and n_{01} represents the number of pairs that are in the same cluster in S. Minkowski Score (MS) is then defined as Eq. (16):

$$MS = \sqrt{\frac{n_{01} + n_{10}}{n_{11} + n_{10}}} \tag{16}$$

Jaccard Score (JS) is another measure of the efficiency of clustering algorithm, whose definition is:

$$JS = \frac{n_{11}}{n_{11} + n_{01} + n_{10}} \tag{17}$$

The third metric called Cluster accuracy (CA) is given as:

$$CA = \frac{1}{N} \sum_{i=1}^{T} max\ Confusion\,(i,j), \quad (i = 1, \cdots T; j = 1, \cdots, S) \tag{18}$$

Table 1
Description of synthesized datasets.

Data set	Number of samples	Data-dimension	Number of clusters
AD_9_2	450	2	9
AD_20_2	1000	2	20
data12	800	2	4
sizes5	1000	2	4
data8	600	2	2
eyes	300	2	3
lineblobs	266	2	3
spiral	1000	2	2

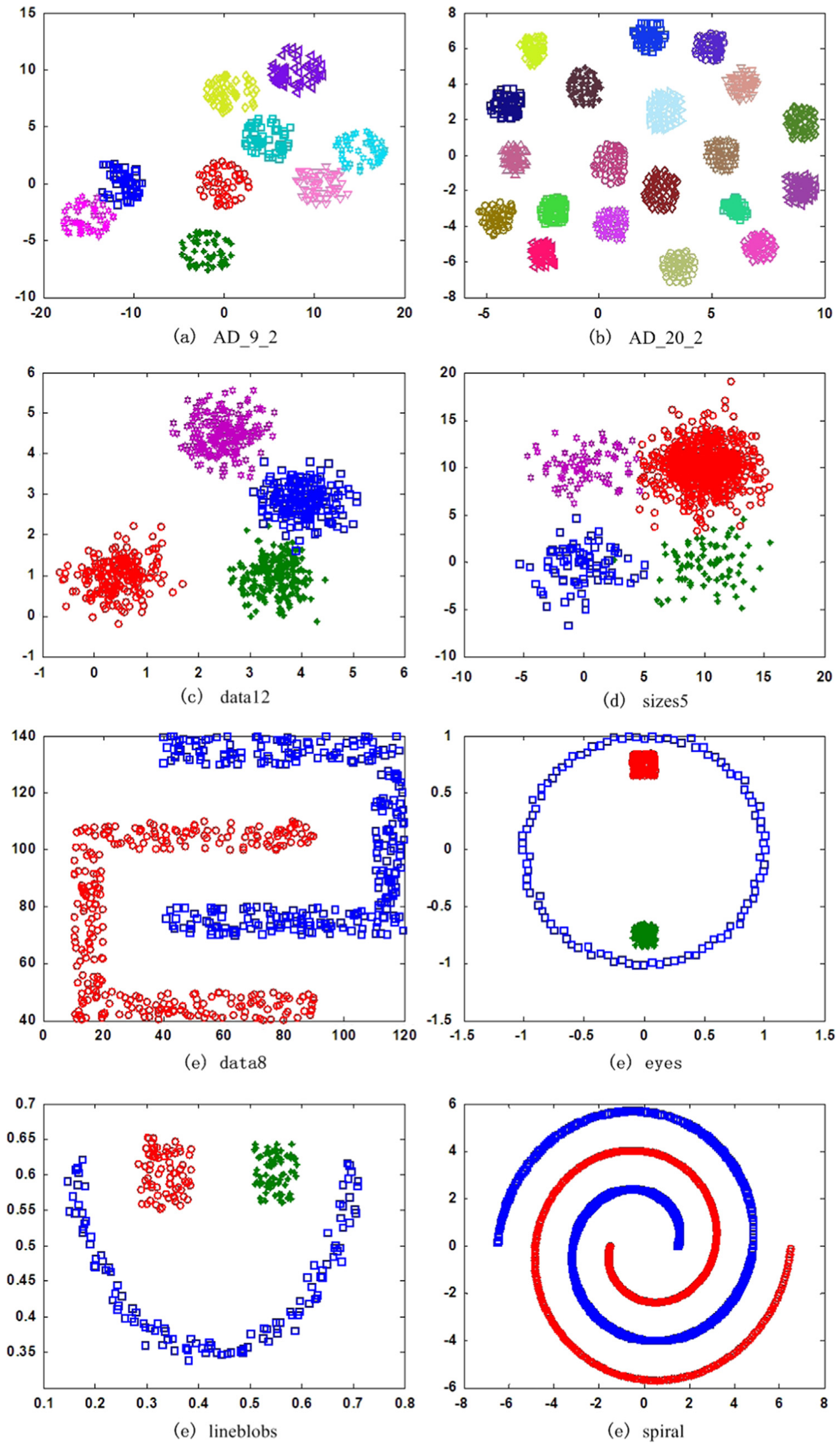

Fig. 3. Synthesized datasets.

where N represents the total number of points in the data set. In this equation, T is the real number of clusters and S is the number of clusters obtained by those algorithms. $Confusion$ represents the confusion matrix, and $Confusion\ (i,j)$ denotes the total number of samples occurring in both j-th cluster obtained by clustering analysis and its corresponding i-th true cluster.

All the metrics return values in the interval [0, 1]. For the completely correct clustering, Minkowski Score is 0, and the two other metrics are 1.

4.2. Experimental results and analysis

For each dataset, the MS, JS and CA or their averages of clustering result are given in Tables 4 and 6. The numbers appeared as bold and red in these two tables denote the best results in all 5 algorithms.

4.2.1. Synthesized datasets

The cluster numbers obtained by KECA-QC on the synthesized datasets are reported in Table 3 firstly. The left side of the slash is the number from KECA-QC. The right side is the true number of dataset. Although the proposed method need no priori knowledge of the cluster number. It can obtain the proper cluster number on all the synthesized datasets. In Table 4, comparing the results by KM with those of KKM, the latter behaves better than KM on all of those synthesized datasets expect for size5, especially on two datasets with manifold structure (eyes and lineblobs). KKM replaces Euclidean distance with Gassion kernel distance. This replacement will alter the dimension structure of data. Therefore, KMM would be more applicable than KM on those datasets with manifold structure, like eyes and lineblobs. But when comparing with NJW, which also have a strong non-linear processing capability, KKM fails to find structures of spiral, while NJW reaches the correct result. More specifically, QC gets the best clustering results on the first three data sets, and a second-best result on size 5. However, this method gets the same poor performance on the manifold datasets and even do not find the data structure as KM. With the preprocessing stage, NJW, KECA-KM and KECA-QC made a feature extraction on initial data, so they can be called spectral-based clustering methods in a broad sense. These algorithms are significantly better than KM and QC on manifold data clustering. KECA-KM gives better clustering results on most data sets compared to NJW. KECA-QC performs best and obtains optimal values on 7 synthesized datasets among these three algorithms. Only for data12, KECA-QC gets sub-optimal results (MS: 0.172, JS: 0.971, CA: 0.993) which are very close to the optimal result obtained by KECA-KM. Note that on 4 manifold datasets, KECA-QC reaches the completely correct results (MS: 0, JS: 1, CA: 1). NJW is completely correct on spiral, and KECA-KM on lineblobs. In addition, the performance of QC and KECA-QC is at approximately the same level on sphere data and dispersive data, while our algorithm enhances the processing capability on manifolds.

Table 2
Description of UCI datasets.

Data set	Number of samples	Data-dimension	Number of clusters
breastcancer(WBC)	683	9	2
german	1000	24	2
glass	214	9	6
heart	270	13	2
ionosphere	351	33	2
iris	150	4	3
new_thyroid	215	5	3
sonar	208	60	2
vote	435	16	2
zoo	101	16	7

Table 3
The cluster number obtained by KECA-QC on synthesized datasets.

Data set	AD_9_2	AD_20_2	data12	sizes5	data8	eyes	lineblobs	spiral
KECA-QC	9/9	20/20	4/4	4/4	2/2	3/3	3/3	2/2

Table 4
Comparison of five algorithms on synthesized datasets.

Data set		KM	KKM	NJW	QC	KECA-KM	KECA-QC
AD_9_2	MS	0.544	0.429	0.497	**0.094**	**0.094**	**0.094**
	JS	0.743	0.813	0.781	**0.991**	**0.991**	**0.991**
	CA	0.871	0.909	0.897	**0.998**	**0.998**	**0.998**
AD_20_2	MS	0.584	0.553	0.516	**0**	**0**	**0**
	JS	0.732	0.753	0.776	**1**	**1**	**1**
	CA	0.863	0.878	0.891	**1**	**1**	**1**
data12	MS	0.286	0.242	0.210	**0.157**	**0.157**	0.172
	JS	0.895	0.922	0.932	**0.976**	**0.976**	0.971
	CA	0.945	0.962	0.964	**0.994**	**0.994**	0.993
sizes5	MS	0.424	0.736	0.224	0.180	0.234	**0.167**
	JS	0.777	0.468	0.946	0.969	0.948	**0.972**
	CA	0.951	0.900	0.960	0.985	0.913	**0.989**
data8	MS	0.739	0.737	0.727	0.757	0.586	**0**
	JS	0.571	0.573	0.582	0.556	0.707	**1**
	CA	0.837	0.838	0.843	0.827	0.905	**1**
eyes	MS	0.844	0.100	0.150	0.895	0.372	**0**
	JS	0.523	0.945	0.921	0.502	0.871	**1**
	CA	0.753	0.963	0.943	0.697	0.963	**1**
lineblobs	MS	0.881	0.017	0.109	0.908	**0**	**0**
	JS	0.449	0.990	0.940	0.433	**1**	**1**
	CA	0.741	0.994	0.964	0.714	**1**	**1**
spiral	MS	0.983	0.972	**0**	0.984	0.813	**0**
	JS	0.349	0.359	**1**	0.347	0.503	**1**
	CA	0.592	0.612	**1**	0.588	0.791	**1**

Table 5
The cluster number obtained by KECA-QC on UCI datasets.

Data set	breastcancer(WBC)	german	glass	heart	ionosphere
num	2/2	23/2	5/6	2/2	2/2
Data set	iris	new_thyroid	sonar	vote	zoo
num	3/3	3/3	2/2	2/2	9/7

4.2.2. UCI datasets

In Table 5, the cluster number obtained by KECA-QC on the UCI datasets are presented. The proposed method obtain the proper cluster number on most of the instance. But with the dataset german the KECA-QC has a poor performance on the cluster number. Table 6 reports the comparison results on metrics with different algorithms. KECA-QC performs significantly better on 6 out of the total 10 UCI datasets and KECA-KM obtains the optimal results on the other 4 groups of data. Meanwhile, experimental results show that KECA-QC outperforms KM, KKM, NJW and the traditional QC in all datasets. KECA-KM gets suboptimal results primarily because it uses *k*-means method based on cosine angle distance to identify clusters, and the component of KECA often lead to a dataset with an angular structure [6]. Overall, the advantages of KECA-QC on UCI data are not as evident as on synthesized datasets. This is mainly due to the fact that the structure of UCI data is not so simple as that of synthesized datasets and it is really complex with great diversity. Moreover, the attributes of UCI data are more complex. In summary, the improved algorithm still achieves better results than the other 4 algorithms.

4.3. Contribution of operations on the clustering performance

4.3.1. KECA operation

In Tables 4 and 6, experimental results demonstrate that KECA-QC obviously outperforms the traditional QC. It is primarily due to the extraction of useful information by using kernel entropy component analysis. Because of the consistency of these three metrics, we use CA as the only evaluation index to make a comparison between KECA-QC and the traditional QC which is preprocessed by SVD. The results are shown in Fig. 4.

In Fig. 4, the natural numbers 1, 2, …, 8 along the horizontal axis denote the used 8 synthesized datasets, and the last 10 numbers represent UCI datasets.

As illustrated in Fig. 4, QC with KECA obtains better performance than this algorithm with SVD for most datasets (12 datasets). Besides, QC (with KECA) shows the same results as QC (with SVD) on five data sets. Therefore, KECA preprocessing stage can help to get better clustering performance, especially on the manifold datasets (5, 6, 7, 8).

4.3.2. *K*-nearest neighbors operation

Furthermore, in order to determine the contribution of *K*-nearest neighbors, a comparison of KECA-QC with and without *K*-nearest neighbors is made. Table 7 shows the CA and running time (seconds) of them.

Table 6
Comparison of five algorithms on UCI datasets.

Data set		KM	KKM	NJW	QC	KECA-KM	KECA-QC
Breastcancer (WBC)	MS	0.373	0.338	0.331	0.475	**0.298**	0.323
	JS	0.871	0.891	0.896	0.792	**0.914**	0.900
	CA	0.961	0.968	0.969	0.934	**0.975**	0.971
german	MS	0.870	0.928	0.844	0.896	**0.729**	0.880
	JS	0.491	0.368	0.571	0.423	**0.622**	0.461
	CA	0.700	0.700	0.709	0.700	**0.810**	0.700
glass	MS	1.093	1.045	1.037	1.086	**0.992**	1.084
	JS	0.318	0.273	0.278	0.245	**0.286**	0.342
	CA	0.569	0.570	0.591	0.561	**0.640**	0.584
heart	MS	0.977	0.992	0.974	0.964	0.947	**0.945**
	JS	0.363	0.338	0.356	0.390	0.438	**0.420**
	CA	0.593	0.556	0.600	0.622	0.652	**0.656**
ionosphere	MS	0.871	0.855	0.879	0.947	0.871	**0.562**
	JS	0.436	0.452	0.428	0.391	0.436	**0.736**
	CA	0.712	0.730	0.704	0.641	0.712	**0.906**
iris	MS	0.662	0.635	0.623	0.583	0.567	**0.320**
	JS	0.658	0.678	0.682	0.717	0.724	**0.903**
	CA	0.848	0.865	0.879	0.900	0.907	**0.973**
new_thyroid	MS	0.666	0.701	0.847	0.628	0.604	**0.600**
	JS	0.632	0.592	0.578	0.708	0.715	**0.724**
	CA	0.843	0.826	0.754	0.865	0.874	**0.875**
sonar	MS	0.992	0.992	0.991	0.991	**0.913**	0.946
	JS	0.340	0.341	0.355	0.357	**0.464**	0.387
	CA	0.553	0.549	0.557	0.558	**0.702**	0.659
vote	MS	0.669	0.656	0.665	0.663	0.653	**0.638**
	JS	0.630	0.640	0.631	0.633	0.642	**0.656**
	CA	0.862	0.870	0.866	0.867	0.871	**0.878**
zoo	MS	0.783	0.743	0.820	0.827	0.489	**0.478**
	JS	0.479	0.532	0.423	0.494	0.780	**0.793**
	CA	0.822	0.821	0.782	0.772	0.852	**0.881**

In Table 7, the bold numbers denote the optimal values between these two comparisons. A slight performance gain could be added by *K*-nearest neighbors, and there is a significant reduction in running time. In order to prove its superiority clearly, these results in Table 7 are illustrated with bar charts which are shown in Figs. 5 and 6.

From Table 7 and Fig. 5 together, we can see that over these 18 datasets, the algorithm with *K*-nearest neighbors obtains better CAs than the algorithm without it on 8 datasets, including data12, sizes5, glass, heart, iris, sonar, vote, zoo. But the improvement is not obvious. These two compared methods obtain the same results on 10 datasets, among which they both get the completely correct clustering results on the manifold datasets. Only on breastcancer (WBC), the algorithm with *K*-nearest neighbors obtains a worse result than the algorithm without *K*-nearest neighbors. Through the above analysis, it indicates that there is little difference between the two algorithms. In most cases, the first *K*-nearest neighbors are enough to express the structure of the data and the redundant data information may not play a positive role in clustering. In some cases, too much information can even reduce the cluster accuracy. Among all the experiments, only the result on breastcancer (WBC) is unsatisfactory. It is mainly due to that we fail to extract all the valid information of the data by using only 50-nearest neighbors of 683 data points. In other words, 50-nearest neighbors are not enough for breastcancer (WBC), and

Table 7
Influence of *K*-nearest neighbors.

Data set	KECA-QC(no *K*-nearest)		KECA-QC(*K*-nearest)	
	CA	Time(s)	CA	Time(s)
AD_9_2	**0.998**	2.89	**0.998**	**1.84**
AD_20_2	**1**	24.4	**1**	**5.92**
data12	0.991	6.27	**0.993**	**3.52**
sizes5	0.985	11.7	**0.989**	**6.11**
data8	**1**	3.66	**1**	**1.73**
eyes	**1**	0.92	**1**	**0.78**
lineblobs	**1**	0.75	**1**	**0.63**
spiral	**1**	10.5	**1**	**7.59**
breastcancer(WBC)	**0.972**	3.61	0.971	**2.25**
german	**0.700**	8.75	**0.700**	**4.36**
glass	0.579	0.67	**0.584**	**0.58**
heart	0.652	0.70	**0.656**	**0.63**
ionosphere	**0.906**	1.03	**0.906**	**0.88**
iris	0.900	0.48	**0.973**	**0.28**
new_thyroid	**0.875**	0.55	**0.875**	**0.52**
sonar	0.605	0.49	**0.659**	**0.47**
vote	0.874	1.49	**0.878**	**1.06**
zoo	0.871	**0.30**	**0.881**	0.38

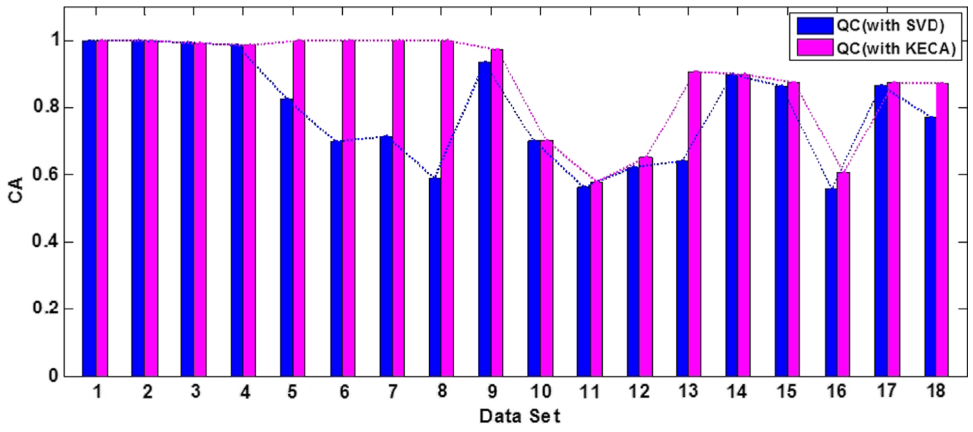

Fig. 4. Comparison between SVD and KECA.

that is why the cluster accuracy is decreased. If we select more neighbors, the clustering result will be better.

The total running time of the compared algorithms is shown in Table 7 and Fig. 6. It can be seen that the running time of the algorithm with K-nearest neighbors is less than the algorithm without it on these data sets except zoo. The more the data points, the more obvious the effect of the algorithm with K-nearest neighbors is. For those data sets which contain more than 500 data points, the time reduction is quite clear in Fig. 6. For example, the execution time of AD_20_2 is reduced from 24.4 s to 5.92 s and from 8.75 s to 4.36 s on german. In addition, the CAs of them remain the same. When the number of the data points is less than 500, the running time is also decreased, while it does not perform significantly better than that on large-scale data sets as we can see in Fig. 7. And, indeed, the less the number of samples is, the less the time requires. For the zoo dataset, the algorithm with K-nearest neighbors requires 0.078 more time than it without K-nearest neighbors. It is because there are only 101 samples in total, and the time spent on finding these points' K-nearest neighbors is larger than estimating their wave functions by using all of the data points. Quantum clustering with K-nearest neighbors will help us to cope with large-scale data.

4.4. Stability and robustness analysis

For analyzing the stable performance of KECA-QC algorithm, the synthesized datasets and UCI datasets mentioned above are chosen. mean and standard deviation are used as judging criterias. Since QC and KECA-QC are both deterministic algorithms, the standard deviations obtained by our algorithm are always zero if the experiments are repeated on the same data. It can be say that our method has greater stability than other stochastic clustering algorithms. In the following experiments, we repeatedly select subsets of the data samples, cluster them, and then see for each result. A distribution over the scores is obtained, so we can compute mean and standard deviation for statistical tests. Table 8 shows mean and standard deviation CA values of six algorithms on 20 subsets. As Table 8 shown, the statistical CA values obtained by KECA-QC IS obviously better than the other five algorithms, although some of the scores are not the best. We come to the conclusion that the stable performance of KECA-QC is better than other compared algorithms.

To compare the robustness of these five algorithms, we follow the criteria used in Ref. [33]. The robustness of the algorithm m on a particular dataset $t$can be defined as the ratio (b_{mt}) of the mean value of its adjusted rand index (ARI) [34] to the largest mean value of ARI among all the compared algorithms:

$$b_{mt} = \frac{R_{mt}}{\max\limits_k R_{kt}} \tag{19}$$

where R represents adjusted rand index, which is given as:

$$R(T,S) = \frac{\sum_{i,j}\binom{n_{ij}}{2} - \left[\sum_i\binom{n_{i\cdot}}{2}\cdot\sum_j\binom{n_{\cdot j}}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_i\binom{n_{i\cdot}}{2}+\sum_j\binom{n_{\cdot j}}{2}\right]-\left[\sum_i\binom{n_{i\cdot}}{2}\cdot\sum_j\binom{n_{\cdot j}}{2}\right]\Big/\binom{n}{2}} \tag{20}$$

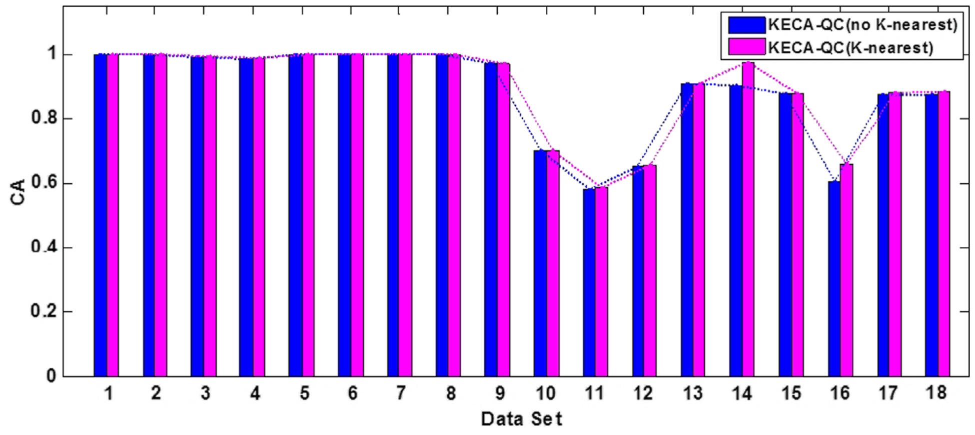

Fig. 5. Influence of K-nearest neighbors on the CA.

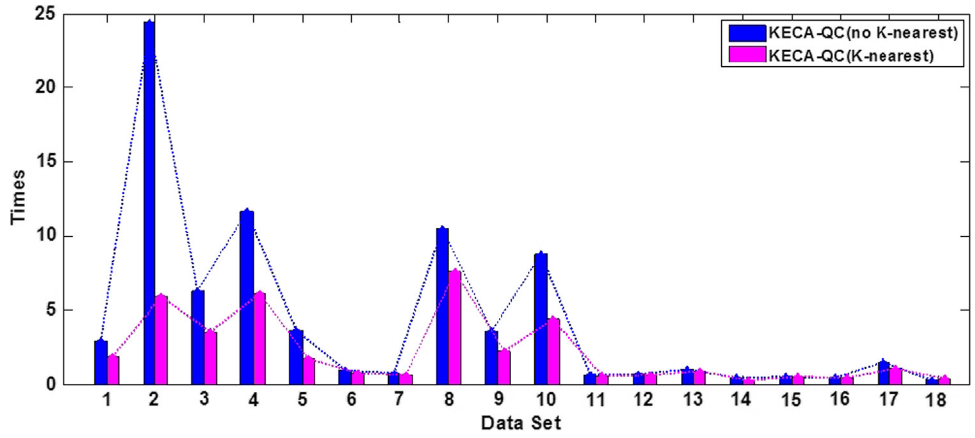

Fig. 6. Influence of K-nearest neighbors on the running time.

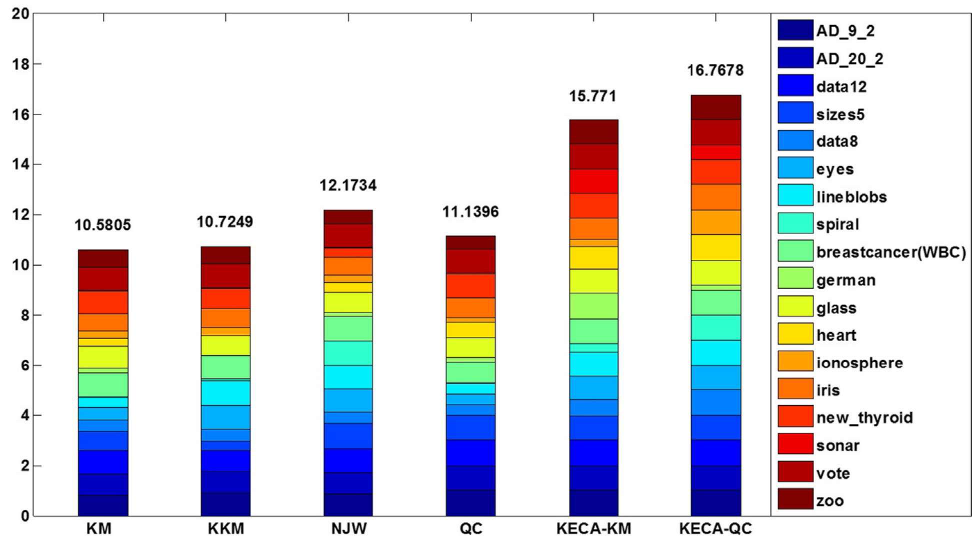

Fig. 7. Comparison of robustness of those six algorithms.

Table 8
Stability analysis on synthesized and UCI datasets.

Data set		KM	KKM	NJW	QC	KECA-KM	KECA-QC
AD_9_2	Mean	0.865	0.827	0.881	0.957	0.958	**0.993**
	Std.	7.14E-03	4.47E-02	1.01E-02	2.35E-02	2.25E-02	**6.45E-03**
AD_20_2	Mean	0.866	0.905	0.894	0.915	0.93	**0.956**
	Std.	1.70E-02	1.28E-02	6.97E-03	5.47E-02	3.70E-02	**3.27E-02**
data12	Mean	0.937	0.9587	0.951	0.948	0.942	**0.995**
	Std.	1.65E-02	8.87E-03	8.69E-03	3.81E-02	5.71E-03	**3.10E-03**
sizes5	Mean	0.93	0.9	0.962	0.97	0.913	**0.982**
	Std.	4.41E-03	6.52E-03	**1.03E-02**	2.63E-02	1.78E-02	1.47E-02
data8	Mean	0.848	0.828	0.845	0.824	0.897	**0.997**
	Std.	1.40E-02	1.35E-02	1.23E-02	2.70E-02	1.91E-02	**1.51E-03**
eyes	Mean	0.729	0.945	0.921	0.705	0.978	**0.999**
	Std.	2.18E-02	2.69E-02	2.35E-02	2.30E-02	2.14E-02	**1.13E-03**
lineblobs	Mean	0.778	0.919	0.919	0.756	0.975	**0.998**
	Std.	2.56E-02	4.39E-02	2.58E-02	2.54E-02	4.95E-03	**1.27E-03**
spiral	Mean	0.615	0.609	0.959	0.587	0.805	**0.979**
	Std.	1.45E-02	**1.32E-02**	7.54E-02	1.79E-02	4.59E-02	3.87E-02
breastcancer (WBC)	Mean	0.966	0.958	0.958	0.933	0.979	**0.987**
	Std.	6.55E-03	5.48E-03	8.83E-03	5.65E-03	**2.77E-03**	9.51E-03
german	Mean	0.692	0.728	0.719	0.73	**0.823**	0.75
	Std.	**8.45E-03**	2.56E-02	1.70E-02	2.85E-02	1.74E-02	3.80E-02
glass	Mean	0.6	0.603	0.57	0.557	**0.653**	0.6
	Std.	2.66E-02	2.11E-02	4.23E-02	4.12E-02	**2.13E-02**	2.25E-02
heart	Mean	0.616	0.5735	0.597	0.634	0.641	**0.659**
	Std.	2.50E-02	**1.17E-02**	2.49E-02	2.70E-02	2.26E-02	2.13E-02
ionosphere	Mean	0.746	0.726	0.709	0.658	0.721	**0.912**
	Std.	2.50E-02	1.08E-02	2.26E-02	2.36E-02	1.75E-02	**6.05E-03**
iris	Mean	0.878	0.85	0.872	0.895	0.903	**0.987**
	Std.	2.94E-02	2.20E-02	6.30E-02	5.02E-02	2.76E-02	**1.27E-02**
new_thyroid	Mean	0.821	0.839	0.761	0.855	0.869	**0.873**
	Std.	3.30E-02	4.80E-02	2.47E-02	4.33E-02	2.53E-02	**2.09E-02**
sonar	Mean	0.605	0.546	0.598	0.567	**0.687**	0.665
	Std.	3.16E-02	1.49E-02	4.64E-02	3.47E-02	**1.23E-02**	1.83E-02
vote	Mean	0.864	0.861	0.876	0.883	0.872	**0.899**
	Std.	1.63E-02	2.54E-02	8.13E-02	4.33E-02	2.14E-02	**1.53E-02**
zoo	Mean	0.822	0.845	0.822	0.803	0.843	**0.889**
	Std.	3.82E-02	2.33E-02	4.32E-02	2.57E-02	2.05E-02	**1.98E-02**

where T, S, i and j are shown in Eq. (18). The ARI returns values in the range of [0, 1].

From Eq. (17), the algorithm m gets the best ARI on the given dataset i when $b_{mi}=1$, and all the other algorithms have $b_{*i} \le 1$. Thus the sum of the ratios over all datasets is represented as b_m which provides a good way to measure the robustness of the algorithm m. The larger value of the sum, the better robustness the algorithm gets.

Fig. 7 shows the distribution of b_m of each algorithm over the 18 datasets. It reveals that KECA-QC has the highest sum value (16.7678). While KECA-KM gets the second-best result (15.771). But, KECA-KM requires providing the number of clusters of datasets. If we just consider the clustering algorithms (QC and KECA-QC) without a-priori knowledge about the number of clusters, KECA-QC considerably improves the performance of clustering. In addition, KECA-QC is more robust than other algorithms (KM, NJW and KECA-KM) which require to know the number of clusters in advance.

4.5. *Test performance on large-scale data*

In this subsection, the performance on datasets with high dimension or large volume, which are two main characteristics of the large-scale data, will be shown and discussed.

Firstly we concern on the datasets with high dimension. In the next experiment, we generate some data sets randomly. Each data set consists of 1000 data points. The data-dimensions are 10, 50, 100, 500, 1000, 5000 and 10,000 respectively. There was no sense in calculating the clustering accuracies because of the random data. Only the statistics of each running time is recorded. Table 8 shows run times averaged over 10 data sets for each value of dimension.

As can be seen from Table 9, the running time increases with the increasing of data-dimension from 10 to 10,000. Even so, the run-time is not more than 30 s. Eq. (9) maps the original dataset to a kernel space. The distance of any pair of samples in the kernel space should be measured. The larger the original data-dimension is, the higher the computational complexity of the kernel matrix generation is. And the data-dimension is not used in the following steps. Therefore, the original data-dimension only affects the calculation of kernel matrix. For example, in this experiment, the running times of quantum clustering stage are approximately 1.5 s, no matter whether the data-dimension is high or not. In order to test the efficiency of KECA-QC on high dimension datasets clustering, we choose 4 real data sets. Table 10 gives attributes and clustering results of datasets. l is the dimension of transformed data after preprocessing. From Table 10, all the datasets reduce their dimensions. The dimension of the lung dataset is reduced from 12,600 to 8, which is the biggest reduction. After processing stage, a low-dimension data can represent a far higher original data well. At the same time, this makes it easy to reduce the performance time of quantum clustering stage. Among these 4 datasets, lung takes longest time (6.83 s) with 203 samples and 12,600 dimensions.

Secondly the performance on datasets with different size will be discussed. Table 11 shows run times on 10 random data sets with different data size. Each data has two dimensions. In our study, the pretreatment KECA may spend slightly more time than SVD, while comparing with tradition QC, KECA-QC can greatly shorten run-time on large volume datasets. The running time on 1000 samples with KECA-QC is about 12.5 s, but 27 s with QC. For 5000 points, the times are 1407 s and 8153 s, respectively. It proves that our algorithm can reduce its running time with the introduction of neighborhood information. We can say larger the size of data is, the more time our method saves. But if one data set is consisted with more than 5000 points, the time cost is still expensive for KECA-QC to achieve data clustering. In Table 12, two data sets are used to test the efficiency. Normal7 is a synthesis of 7 spherical clusters. Twonorm is a real dataset with 7400 samples and 20 dimensions. The dimension of Twonorm is reduced from 20 to 4, after preprocessing, but the dimension of Normal7 is increased from 2 to 7. As mentioned above, the dimension may be increased, as is often the case, particularly for those low-dimensional data sets. Despite that, KECA produces transformed data sets advantageous to the clustering analysis.

From above two aspects, we can conclude that KECA-QC is effective on high-dimensional datasets. This method can handle up to tens of thousands of dimensions, and the running speed can be very fast. Although the proposed method has advantages than the traditional QC in dealing with large volume data set, it still need much time, especially when the data size is more than 5000.

Table 9
Running time for each dimension.

Data-dimension	10	50	100	500	1000	5000	10,000
Time(s)	4.13	4.27	4.53	5.03	6.30	15.8	26.9

Table 10
Performance on high dimension datasets.

Data set	Number of samples	Data-dimension	Number of clusters	KECA-QC				
				l	MS	JS	CA	Times (s)
spell	798	72	5	5	0.805	0.539	0.738	3.52
ovarian	54	1536	2	3	0.555	0.737	0.872	0.12
colon	62	2000	2	7	0.523	0.753	0.919	0.17
lung	203	12,600	5	8	0.504	0.785	0.892	6.83

Table 11
Running time for each data size.

Number of samples		100	500	1000	2000	3000	4000	5000
Time(s)	KECA-QC	0.18	2.49	12.5	112	353	507	1407
	QC	0.17	4.23	27.0	470	1882	4097	8153

5. Conclusions

In this paper, quantum clustering using kernel entropy component analysis is introduced. It keeps the strengths of traditional quantum clustering, such as finding the clusters of any shape, detecting the clustering centers based on the underlying information of the data itself and requiring no priori knowledge about the number of clusters, etc. The improved algorithm can be divided into two stages: preprocessing and clustering stage. In the preprocessing stage, effective features are extracted in the kernel space, and the data points belonging to different clusters are separated by using kernel entropy component analysis. Moreover, it reduces the dimension of the high-dimensional data and it provides a sound basis for further quantum clustering analysis. The performance of the clustering algorithm with this preprocessing is greatly improved compared to the classic quantum clustering. The introduction of the K-nearest neighbors to estimate the wave function significantly reduces the computation time. Meanwhile, the introduction of local information can also help improve the quantum clustering accuracy. The proposed algorithm is

Table 12
Performance on large datasets.

Data set	Number of samples	Data-dimension	Number of clusters	KECA-QC				
				l	MS	JS	CA	Times(s)
Normal7	3500	2	7	7	0.1168	0.9865	0.9966	496.7
Twonorm	7400	20	2	4	0.5184	0.7126	0.8362	3785

applicable to high-dimensional data sets, especially to the complex datasets. It leads to satisfactory results both in the accuracy and the running time. However, the improved algorithm has the shortcomings which are inevitable for quantum clustering.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 61272279, 61371201, 61272282, 51509090 and 61203303), the Program for New Century Excellent Talents in University (No. NCET-12-0920), the Program for New Scientific and Technological Star of Shaanxi Province (No. 2014KJXX-45), the Fundamental Research Funds for the Central Universities (Nos. K5051302049, K5051302023, K5051302002 and K5051302028), the Fund for Foreign Scholars in University Research and Teaching Programs (the 111 Project) (No. B07048), EU IRSES project (No. 247619) and National Science and Technology Major Project (No. 2014ZX02005001).

References

[1] J. McQueen, Some methods for classification and analysis of multivariate observations, in: Proceeding of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281–297.
[2] D. Horn, A. Gottlieb, The method of quantum clustering, in: Proceeding of the Advances in Neural Information Processing Systems (NIPS) 14, 2001, pp. 769–776.
[3] D. Horn, A. Gottlieb, Algorithm for data clustering in pattern recognition problems based on quantum mechanics, Phys. Rev. Lett. 88 (1) (2001) 018702.
[4] D. Horn, I. Axel, Novel clustering algorithm for microarray expression data in a truncated SVD space, Bioinformatics 19 (9) (2003) 1110–1115.
[5] N. Nasios, A.G. Bors, Nonparmetric clustering using quantum mechanics, in: Proceedings of the IEEE International Conference on Image Processing, vol.3, 2005 pp. 820–823.
[6] N. Nasios, A.G. Bors, Kernel-based classification using quantum mechanics, Pattern Recognit. 40 (3) (2007) 875–889.
[7] Z.H. Li, S.T. Wang, Improved algorithm of quantum clustering, Comput. Eng. 33 (23) (2007) 189–190.
[8] Z.H. Li, S.T. Wang, Parameter-estimated quantum clustering algorithm, J. Data Acquis. Process. 23 (02) (2008) 211–214.
[9] Y. Zhang, P. Wang, G.Y. Chen, Quantum clustering algorithm based on exponent measuring distance, in: Proceedings of the IEEE International Symposium on KAMW, 2008, pp. 436–439.
[10] S. P. Gou, X. Zhuang, L. C. Jiao, SAR image segmentation using quantum clonal selection clustering, in: Proceedings of the 2nd Asian-Pacific Synthetic Aperture Radar conference, 2009, pp. 817–820.
[11] S.P. Gou, X. Zhuang, Y.Y. Li, C. Xu, L.C. Jiao, Multi-elitist immune clonal quantum clustering algorithm, Neurocomputing 101 (2013) 275–289.
[12] Y.Q. Niu, B.Q. Hu, W. Zhang, M. Wang, Detecting the community structure in complex networks based on quantum mechanics, Phys. A: Stat. Mech. Appl. 387 (2008) 6215–6224.
[13] J. Sun, S. N. Hao, Research of fuzzy neural network model based on quantum clustering, in: Proceedings of the Second International Workshop on Knowledge Discovery and Data Mining, 2009, pp. 133–136.
[14] E. Di Buccio, G.M. Di Nunzio, Distilling relevant documents by means of dynamic quantum clustering, Advances in Information Retrieval Theory, Lecture notes in computer science, vol. 6931, 2011, pp. 360–363.
[15] R. Jenssen, K.E. Hild II, D. Erdogmus, J.C. Principe, T. Elotoft, Clustering using Renyi's entropy, in: Proceedings of the International Joint Conference on Neural Networks vol.1, 2003, pp. 523–528.
[16] R. Jenssen, Kernel entropy component analysis, IEEE Trans. Pattern Anal. Mach. Intell. 32 (5) (2010) 847–860.
[17] S. Gasiorowicz, Quantum physics, Wiley, New York, 1996.
[18] D. Horn, Clustering via Hilbert space, Physica A 302 (1–4) (2001) 70–79.
[19] Bernhard Schölkopf, Alexander Smola, Klaus-Robert Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319.
[20] Heiko Hoffmann, Kernel PCA for novelty detection, Pattern Recognit. 40 (3) (2007) 863–874.
[21] T. Xiang, S.G. Gong, Spectral clustering with eigenvector selection, Pattern Recognit. 41 (3) (2008) 1012–1029.
[22] F. Zhao, L.C. Jiao, H.Q. Liu, X.B. Gao, M.G. Gong, Spectral clustering with eigenvector selection based on entropy ranking, Neurocomputing 73 (10–12) (2010) 1704–1717.
[23] I. S. Dhillon, Y. Q. Guan, B. Kulis, Kernel k-means: spectral clustering and normalized cuts, in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge discovery and data mining, 2004, pp. 551–556.
[24] J.B. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[25] Y. Ng, M.I. Jordan, Y. Weiss, On spectral clustering: analysis and an algorithm, in: Proceedings of the Advances in Neural Information Processing Systems, 2002, pp. 849–856.
[26] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, Wiley, New York, 2000.
[27] UC Irvine machine learning repository. ⟨http://archive.ics.uci.edu/ml/datasets.html⟩.
[28] A. Ben-Hur, I. Guyon, Detecting stable clusters using principal component analysis, Methods Mol. Biol. 224 (2003) 159–182.
[29] B.W. Silverman, Density estimation for statistics and data analysis, Chapman and Hall, New York, 1986.
[30] R. Varshavsky, D. Horn, M. Linial, Clustering algorithms optimizer: a framework for large datasets, Lecture notes in bioinformatics, 2007, pp. 85–96.
[31] A.G. Bors, N. Nasios, Kernel bandwidth estimation for nonparametric modeling, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 39 (6) (2009) 1543–1555.
[32] 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, no. 281–297, 1967, pp. 14.
[33] X. Geng, D.C. Zhan, Z.H. Zhou, Supervised nonlinear dimensionality reduction for visualization and classification, IEEE Trans. Syst. Man Cybern. Part B 35 (6) (2005) 1098–1107.
[34] L. Hubert, P. Arabie, Comparing partitions, J. Classif. 2 (1) (1985) 193–218.
[35] E. Parzen, On the estimation of a probability density function and the mode, Ann. Math. Stat. 32 (1962) 1065–1076.

Yangyang Li received the B.S. and M.S. degrees in Computer Science and Technology from Xidian University, Xi'an, China, in 2001 and 2004 respectively, and the Ph.D. degree in Pattern Recognition and Intelligent System from Xidian University, Xi'an, China, in 2007. She is currently a professor with Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China at Xidian University. Her current research interests include quantum-inspired evolutionary computation, artificial immune systems, and data mining.