Multimedia Tools and Applications (2019) 78:33415–33434
https://doi.org/10.1007/s11042-019-08009-x

Weighted adjacent matrix for *K*-means clustering

Jukai Zhou[1] · Tong Liu[1] · Jingting Zhu[1]

Received: 10 March 2019 / Revised: 27 May 2019 / Accepted: 16 July 2019 /
Published online: 25 July 2019

Abstract

K-means clustering is one of the most popular clustering algorithms and has been embedded in other clustering algorithms, e.g. the last step of spectral clustering. In this paper, we propose two techniques to improve previous *k*-means clustering algorithm by designing two different adjacent matrices. Extensive experiments on public UCI datasets showed the clustering results of our proposed algorithms significantly outperform three classical clustering algorithms in terms of different evaluation metrics.

Keywords *k*-means clustering · Similarity measurement · Adjacent matrix · Unsupervised learning

1 Introduction

Clustering is designed to partition a set of data points into groups, where similar data points are in the same groups and dissimilar data points are in different groups [49]. Different from supervised learning, clustering is one of algorithms of unsupervised learning [20, 50] which conducts data analysis without the help of labels. Hence, clustering has been attracting extensive research interests and has been successfully applied in the areas of data mining and machine learning [38]. For example, before constructing a classifier, the literature in [38] first conducts *k*-means clustering to reduce the computation time of the classification task.

Among previous clustering algorithms, *k*-means clustering is a widely used algorithm due to its linear time complexity and ease of implementation. However, *k*-means clustering is limited to its applicability due to the issues, such as identification of the cluster number *k*, initialisation of centroids, as well as the definition of similarity measurements for evaluating the similarity between two data points [24]. In the past years much efforts have been devoted for addressing these issues, such as rule of thumb method [22] and gap statistic method [36] for selecting the optimal value of *k*, hierarchical centroid selection and simple cluster seeking [22]

✉ Tong Liu
t.liu@massey.ac.nz

[1] School of Natural and Computational Sciences, Massey University, Auckland, New Zealand

for centroid initialisation, self-paced learning technique [46] and multiple feature extraction algorithm [1] for constructing the similarity matrix.

Another popular clustering algorithm is spectral clustering, which uses spectral representation (measuring the relationship among data points, as knowns as the high-order relationship [48]) to replace the original representation (as known as low-order relationship) via a two-step strategy, i.e., generation of spectral representation (i.e., similarity matrix learning) followed by conducting *k*-means clustering on the resulting spectral representation. Spectral clustering has also been shown to outperform *k*-means clustering in many kinds of applications, which implies that representation learning is very important for *k*-means clustering [44, 50].

In this paper, based on the above observation, we focus on investigating an effective similarity matrix for addressing the third limitation of *k*-means clustering, i.e., the definition of similarity measurements [47]. With the help of the effective similarity matrix, our proposed method improves the clustering effectiveness. Specifically, inspired from the spectral clustering algorithm, we first design two new representations of original features separately, i.e., an adjacent matrix and a weighted adjacent matrix, to represent the original data points, and then conduct *k*-means clustering on the new representations to output the clustering results.

The rest of this paper is organised as below. In section 2, we briefly introduce *k*-means clustering and spectral clustering algorithms, followed by proposing our methods in Section 3. We then conduct experimental analysis on real UCI datasets for comparing our proposed methods with previous clustering algorithms in Section 4. Finally, in Section 5, we conclude our work, followed by proposing future research work.

2 Related work

In the literature, previous clustering algorithms are partitioned into the following categories, such as partition based clustering algorithms, hierarchy based clustering algorithms, density based clustering algorithms, graph based clustering algorithms, grid based clustering algorithms, and kernel based clustering algorithms.

2.1 Partition based clustering algorithms

The basic idea of partition based clustering algorithms is to identify the centroids of all data points. Specifically, for a given similarity measurement, the similarity between two data points and a centroid are first calculated, and then the similarity is compared with the predefined threshold. Once meeting the criteria, this data point will be classified into the cluster of this centroid. The typical algorithms of partition based clustering include *k*-means clustering and its variants, e.g. *k*-medoids [2] and *k*-means++ [3]. Recently, both balanced *k*-means [27] and recursive partition based *k*-means [6] dramatically reduce the computational complexity for conducting clustering on massive datasets.

2.2 Hierarchy based clustering algorithms

The basic idea of hierarchy based clustering algorithms is to produce a sequence of nested partitions, in which a single cluster is created on the top of all other singleton clusters and all the data points are included at the bottom. In the hierarchy based clustering algorithm, each level in the middle can be deemed as a combination from the lower levels. By this means, the hierarchical clustering algorithm can be graphically demonstrated as a tree, which can be produced in two ways, i.e., divisive and agglomerative. Divisive method is to start with one all-inclusive cluster, and then splits the tree step by step until the similarity among data points within a cluster meets the criteria. Agglomerative method starts with all data points as a single cluster, and then merges the closest cluster pairs. Classic hierarchical clustering algorithms include balanced iterative reducing and clustering using hierarchies (BRICH) [4], clustering using representatives (CURE) [18] and robust clustering using links (ROCK) [19] . However, most hierarchical clustering algorithms are sensitive to noise, indicating that the clustering result may be affected by even few minor outliers [28]. Hence, some enhanced hierarchical clustering algorithms, e.g., robust hierarchical *k*-center clustering [23], are developed to address this issue.

2.3 Density based clustering algorithms

The most important idea of density based clustering algorithms is that there should be enough neighbouring data points for each data point in a cluster under a designated similarity measurement. In this case, the data point without meeting the threshold will be regarded as noise, and will not belong to any cluster. Density based clustering algorithms can be used to partition arbitrary shapes as long as the target clusters have different density. Density-based spatial clustering of applications with noise (DBSCAN) [16] and ordering points to identify the clustering structure (OPTICS) [10] are the conventional representatives of density based clustering algorithms, while influence space DBSCAN [7] and DBSCAN based on influence space and detecting of border points [26] are their revised versions. Most recently, RNN-DBSCAN [5] uses the number of reverse nearest neighbours as an estimate of observation density, while *k*-nearest neighbor DBSCAN [31] uses *k*-nearest neighbour representatives for density based clustering without parameters pre-definition. In nutshell, the recent developed density based clustering algorithms are more efficient and effective than conventional DBSCAN and OPTICS algorithms.

2.4 Graph based clustering algorithms

The key idea of graph theory based clustering algorithms is to build a similarity matrix (i.e., graph) using all training data, and then uses this graph to generate a new representation of the original data points to conduct clustering. Since the graph based clustering algorithm takes into account the similarity relationship, i.e., replacing the original data points by high-order relationship representation [45] [43]. Hence, the clustering process is indeed finding a solution of optimal graph cutting, which is able to achieve higher efficiency than other clustering algorithms. However, graph based clustering algorithms are usually with high computation complexity (i.e., at least quadratic to the sample size) due to the construction of the high-order relationship representation. Cluster identification via connectivity kernels (CLICK) [40] is a classic representative of graph based clustering algorithms which aims to find out the minimum weight division of the graph literately. Other graph based clustering algorithms include structural clustering algorithm for networks (SCAN) [41], SCAN++ [32], pruned SCAN (pSCAN) [8] and Scalable Density-Based Graph Clustering (ScaleSCAN) [33].

The most famous and popular graph based clustering algorithm is spectral clustering. Due to excellent characteristics of resilience and high efficiency, a wide range of spectral clustering variants have been developed, such as low-rank sparse subspace spectral clustering [49], fast large-scale spectral clustering via explicit feature mapping [20], and one-step multi-view spectral clustering [50].

2.5 Grid based clustering algorithms

Grid based clustering algorithms focus on searching a space surrounding the data points and excluding the data point itself only. To do this, a grid structure is constructed with a finite number of cells, in which the data points will be mapped and partitioned. Specifically, the centroid will be identified by computing the density of each cell and sorting the cells by different densities. During the whole clustering process, all the calculations are operated on grid cells and nothing is done with the data points themselves. For example, statistical information gird (STING) [35] takes advantage of both grid clustering algorithm and parallel computing. Recently, a novel grid based clustering algorithm for hybrid data stream (FGCH) [9] is designed for dealing with hybrid data, while the improved grid-based clustering algorithm with diagonal grid searching and merging (DSM) [25] is an improved version of grid-based clustering algorithm with diagonal grid searching and merging.

2.6 Kernel based algorithms

The key idea of kernel based algorithms is to create a high-dimensional feature space, in which the data points with non-linear relationship are able to be linearly partitioned. Actually, in order to firstly map non-linear data structure to linear space and then apply conventional clustering algorithms, kernel based clustering algorithms are often used with other clustering algorithms together. For example, kernel *k*-means clustering combines the kernel based algorithm with *k*-means clustering algorithm, while kernel-based fuzzy c-means clustering [11] combines the conventional fuzzy c-means clustering algorithm with kernel resolution to take advantage of genetic algorithm. Recently, the kernel-based hard clustering algorithm in [15] and the robust multiple kernel *k*-means clustering [13] have been shown to be able to improve clustering performance significantly by using kernel theory.

3 Methods

3.1 Preliminary

To help understand our methods, we firstly introduce the fundamentals and implementation of *k*-means clustering and spectral clustering.

3.1.1 *k*-means clustering

Generally speaking, *k*-means clustering is designed to group a set of data points into *k* clusters where the data points in the same cluster have maximal similarity while the data points among different clusters have maximal dissimilarity. In this paper, we first let $\mathbf{X} = \{\mathbf{x}_1,\ldots,\mathbf{x}_n\}\in \mathrm{R}^{n\times d}$ be the input data points, $\mathbf{x}_i$ be the *i*-th row of **X,** and $x_{i,j}$ be the element of *i*-th row and *j*-th column in matrix **X**, and then describe the brief implementation of *k*-means clustering in Table 1 below.

Table 1 The pseudo code of *k*-means clustering

```
Input: data points $X = \{x_1,\ldots,x_n\} \in R^{n \times d}$; the cluster number $k$.
Output: the cluster indicators of all data points and centroids C.
1: Centroid initialisation by randomly selecting $k$ data points;
2: do
3: Assign data points to the closest centroids to form $k$ clusters;
4: Update each centroid by the mean value of data points within each cluster;
5: until
6: Algorithm converges and centroids have no changes.
```

Actually, the goal of *k*-means clustering is to achieve the minimum sum-squared-error (SSE), which means the minimal total intra-cluster variance by a given *k*:

$$\mathrm{SSE} = \sum_{j=1}^{k} \sum_{i=1}^{\mathrm{t}_j} \left\| \mathbf{x}_i - \mathbf{c}_j \right\|_2^2 \tag{1}$$

Where k denotes the number of clusters, t_j denotes the number of data points in the j-th cluster, and c_j denotes the centroid of the j-th cluster. $\|\mathbf{x}_i - \mathbf{c}_j\|_2$ denotes the l_2 norm of $\mathbf{x}_i - \mathbf{c}_j$. Usually, due to the randomness of centroids selection, the clustering result with the minimal SSE may achieve a local optimal result, so the initial centroids will put a significant influence on the clustering result. Besides, both predicting the actual cluster number and defining the similarity measurements are also major issues of *k*-means clustering. We will list these issues in details as follows.

In real applications, the actual cluster number *k* is always unknown and there is no efficient solution in theory to identify the value of *k*, so a number of literatures have focused on solving

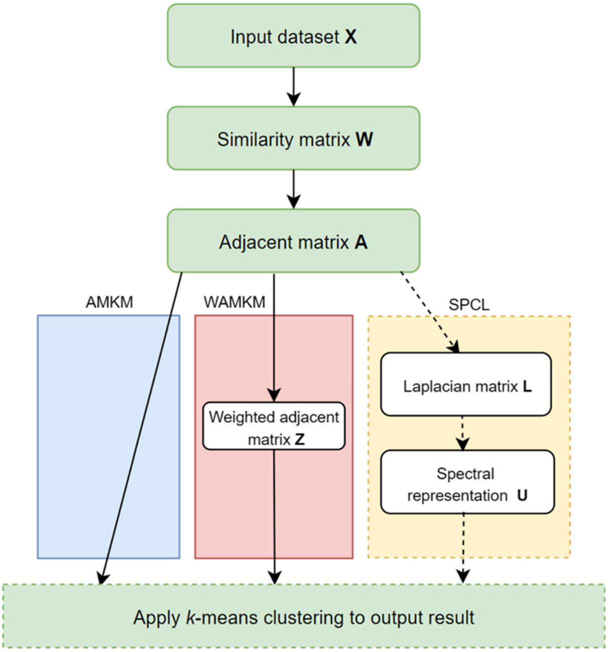

Input dataset X
Similarity matrix W
Adjacent matrix A
AMKM
WAMKM
SPCL
Weighted adjacent matrix Z
Laplacian matrix L
Spectral representation U
Apply k-means clustering to output result

Fig. 1 The graphical structures of our proposed methods (left and middle) and spectral clustering (SPCL). It is noteworthy that green parts are common for all three methods

this issue. For example, on-demand selection algorithm manually selects the value of k as the actual cluster number. Elbow method determines the value of k based on the vision of the SSE-k graph, and the gap statistic method can be regarded as a revised version of Elbow method. Besides, the rule of thumb method designs the following equation to obtain the value of k:

$$k \approx \sqrt{\frac{n}{2}} \tag{2}$$

Another problem of k-means clustering is to identify the initial position of centroids. The simplest way is to choose the initial centroids at random. However, the experiment results indicate that the random initialisation puts a significant effect on the final clustering result, and even causes bad or complete wrong partitions. Recently, some effective solutions were developed to address this issue. For example, the hierarchical centroid selection [42] first runs basic k-means clustering multiple times with random initialisation so that a group of centroids will be produced, then this group of centroids will be regarded as input data points to carry out final centriods. Simple cluster seeking (SCS) [30], which is the default algorithm for k-means clustering in Matlab software suite, selects the first centroid at random and marks it as k_1, and then finds out the next data point with the maximal distance to k_1 as the second centroid k_2. This process is repeated until k centroids are generated.

The third issue of k-means clustering is to define the similarity measurement between two data points. In other words, a larger distance between two data points means smaller similarity. In real applications, Euclidean distance and its variants are widely used by k-means clustering as similarity measurement, and other similarity measurements including cosine similarity, Jaccard coefficient, Pearson correlation coefficient and averaged Kullback-Leibler divergence are also widely used.

3.1.2 Spectral clustering

Recently, spectral clustering [37] is becoming increasingly popular. Comparing to the traditional k-means clustering, spectral clustering pre-processes training data points by replacing low-order relationship or original data points with high-order relationship representation [29, 39]. To achieve this, spectral clustering, firstly, constructs a similarity matrix $\mathbf{W}$, which contains the similarity relationship between every two data points. When the similarity measurement is Euclidean distance, the similarity matrix is defined as:

$$w_{i,j} = \sqrt{\sum_{t=1}^{d} \left(x_{i,t} - x_{j,t}\right)^2} \quad (i, j \in [1, n], t \in [1, d]) \tag{3}$$

Where i and j, respectively, denote the i-th and j-th data point, and t denotes the t-th feature of the data point.

Secondly, spectral clustering transfers the similarity matrix to a sparse matrix by using a kernel function, and then produces the Laplacian matrix $\mathbf{L}$. In this paper, we term this sparse similarity matrix as adjacent matrix $\mathbf{A}$, and then define the normalised Laplacian matrix $\mathbf{L}$ as:

$$\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-\frac{1}{2}} \tag{4}$$

Where $\mathbf{D}$ is a diagonal matrix whose elements are the summation of each row of $\mathbf{A}$ (or column as $\mathbf{A}$ is symmetric), i.e., $d_{i,i} = \sum_{j=1}^{n} a_{i,j}$.

Finally, spectral clustering conducts dimension reduction by selecting k eigenvectors of **L,** and then conducts k-means clustering on the reduced matrix to output the final clustering result. We list the details of spectral clustering in Table 2.

3.2 Our methods

In this section, we develop two novel clustering methods and list the details as follows. Specially, the first method called adjacent matrix based k-means clustering method (AMKM) runs k-means clustering on the adjacent matrix directly, while the second method called weighted adjacent matrix based k-means clustering method (WAMKM) takes into account the weight of the features. We introduce their graphical structures as follows:

3.3 Adjacent matrix based *k*-means clustering method (AMKM)

The first step of the spectral clustering is to construct the similarity matrix by transferring the data points into an undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ denotes the vertices, and $\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_m\}$ $(m = n\times\ (n\text{-}1)/2)$ denotes the edges between vertices. The undirected graph is abstracted and represented by the similarity matrix $\mathbf{W} = \left(w_{i,j}\right)_{i,j=1}^{n}$, where $\mathrm{w}_{i,\,j} \geq 0$ means the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$ under a given distance metric. The adjacent matrix **A** is constructed based on **W** by the following methods.

In the past decades, researchers have paid much effort on constructing the adjacent matrix, including ε-neighbourhood graph, k-nearest neighbour graph, and fully connected graph [39]. For example, the ε-neighbourhood graph connects two neighboured vertices (i.e., $\mathrm{e}_m = 1$) if the pairwise distance is less than a given threshold ε, otherwise, $\mathrm{e}_m = 0$. This makes all edges of a graph roughly have the same value (i.e., ε**)** and leads to an unweighted graph. The k-nearest neighbour graph connects $\mathbf{v}_i$ and $\mathbf{v}_j$ if $\mathbf{v}_j$ is one of k nearest neighbours of $\mathbf{v}_i$, which results in a directed graph due to the asymmetry of neighbourhood relationship, so that additional effort is required to make the graph symmetric. The fully connected graph simply connects all the vertices with the similarity scalar between each other. In this paper, we choose to construct a fully connected graph, so that the most important step of constructing adjacent matrix is to represent the distance between data points by an appropriate similarity function. The widely used kernel functions include Polynomial kernel, Gaussian kernel [21] and Sigmoid kernel. When a Gaussian kernel function is used, the adjacent matrix is defined as follows:

$$a_{i,j} = e^{-\left(\frac{\|w_i - w_j\|_2^2}{2*\sigma^2}\right)} \qquad (i, j \in [1, n]) \tag{5}$$

Table 2 The pseudo code of spectral clustering

```
Input: data points $X = \{x_1,\ldots,x_n\} \in R^{n\times d}$; the cluster number $k$.
Output: the cluster indicators of all data points and centroids C.
1: Compute the similarity matrix W of X by Eq. (3);
2: Compute the Laplacian matrix L by Eq. (4);
3: Compute the first $k$ eigenvectors of L, marked as $E = \{e_1, \ldots, e_k\}$;
4: Construct matrix U, where $U = E^T$, $U \in R^{n\times k}$;
5: Run $k$-means clustering on U to output the cluster result C.
```

After this, the next step of the spectral clustering is to compute the graph Laplacian, and then outputs the first k eigenvectors, which are used as the input of k-means clustering. However, when the dataset is relatively large, the computational complexity is time consuming.

To address this issue, in our first method AMKM, we directly run k-means clustering on the adjacent matrix instead of the Laplacian eigenvector matrix. By this means, we can avoid both the computation cost of the Laplacian matrix and the optimization cost of eigenvalue decomposition. As a result, the computing complexity in AMKM is reduced. This makes it possible to run on large datasets. The details of AMKM is briefly described in Table 3.

3.4 Weighted AMKM (WAMKM)

In the last section, we introduced our improved k-means clustering method. However, in the real world, it is well known that a data point consists of multiple features with different priorities, and it is obvious that different features always put different influence on the clustering result. Generally speaking, an important feature always affects even more on the clustering result than the unimportant features. This means features have different importance or weight [17]. From this perspective, we should give more priority on the feature that has more weight when constructing the adjacent matrix. In this section, we introduce our second clustering method – weighted adjacent matrix based k-means clustering method (WAMKM), which takes the weight of features into account.

For each data point representation in the adjacent matrix **A**, every feature is described by a numeric scalar, so in our paper we calculate the weight by the percentage of each feature among all features. Specifically, we first calculate the summation of all data points for each feature to produce the weight vector **d** ($\mathbf{d} = \{\mathrm{d}_1,\ldots,\mathrm{d}_n\}$), where d_j is the summation of all elements in the j-th column of **A**, and then we normalise the weight vector by:

$$\mathbf{h} = \frac{d_j}{\sum\limits_{j=1}^{n} d_j} \qquad (j \in [1, n]) \tag{6}$$

Eq. (6) makes the sum of all elements in **h** be 1, where every element h_j in the j-th element represents the probability or the contribution of the j-th feature to all data points. In this way, we consider the feature importance. Furthermore, we produce the weighted adjacent matrix **Z** by applying the weight vector **h** on each data point in adjacent matrix **A**:

$$z_{i,j} = a_{i,j} \times \mathrm{h}_j \qquad (i, j \in [1, n]) \tag{7}$$

Table 3 The pseudo code of our proposed AMKM method

```
Input: data points $X = \{x_1,\ldots,x_n\} \in R^{n \times d}$; the cluster number $k$.
Output: the cluster indicators of all data points and centroids C.
1: Calculate the similarity matrix W of X by Eq. (3);
2: Calculate adjacent matrix A by Eq. (5);
3: Run $k$-means clustering on A to output C.
```

Finally, after the weighted adjacent matrix **Z** is produced, we apply *k*-means clustering on it in order to output the clustering result, which is also the clustering result of the original dataset. The steps of WAMKM is briefly described in Table 4.

4 Experiments and analysis

In this paper, we selected twelve datasets to evaluate our two clustering methods, compared with three clustering algorithms, in terms of three clustering evaluation metrics.

4.1 Datasets

The selected twelve datasets are from both UCI Machine Learning Repository and data mining centre website. These datasets belong to different categories and have wide range varieties of characteristics, which are able to fully evaluate the reliability and effectiveness of our proposed methods. We summarise the datasets used with their details in Table 5.

4.2 Comparison algorithms

In this paper, we use the following clustering algorithms as comparison algorithms.

- *k*-means clustering is the most popular and widely used clustering algorithm, which aims to group the data points into *k* clusters where the data points in the same cluster are as similar as possible and data points in the different clusters are as dissimilar as possible. In our implementation, we use the Matlab build-in function, with the "distance" parameter set to "Euclidean distance" and the "initial centroid position selection algorithm" parameter set to "cluster".
- *k*-means++ clustering is a variant of *k*-means clustering which uses a heuristic strategy to find centroids. In some cases, *k*-means++ clustering converges faster and achieves a lower sum of SSE, compared to standard *k*-means clustering algorithm.
- Normalised spectral clustering (SPCL) [39] is a widely used variant of the spectral clustering algorithms. Specifically, it conducts *k*-means clustering on the normalised eigenvector matrix by normalizing the row sum to have the norm of 1.

4.3 Parameters settings

In our experiment, we used 10-fold cross-validation method to evaluate all the algorithms. Our proposed methods need to tune the parameter σ which plays a vital influence on the kernel

Table 4 The pseudo code of our proposed WAMKM method

```
Input: data points $X = \{x_1,\ldots,x_n\} \in R^{n \times d}$; the cluster number $k$.
Output: the cluster indicators of all data points and centroids C.
1: Produce the similarity matrix W of X by Eq. (3);
2: Calculate the adjacent matrix A by Eq. (5);
3: Calculate the weight vector h by Eq. (6);
4: Calculate the weighted adjacent matrix Z by Eq. (7);
5: Run k-means clustering on Z to output C.
```

Table 5 Summary of the datasets used in this paper

Datasets	Samples	Features	Classes
20news	3970	8014	4
Binalpha	1404	320	9
Australian Credit Approval	690	14	2
Website Phishing	1353	9	3
Dexter	300	20000	2
Diabetes	768	8	2
Coil20Data	1440	1024	20
Cardiotocography	2126	41	3
Spambase	4601	57	2
Parkinson Speech	1040	28	2
Solar Flare	1066	12	6
German Credit Data	1000	23	2

function performance and the clustering result [34]. Specifically, in our experiment, we tested the parameter σ in the range of $\sigma\in [10^{-5}, \ldots 10^{14}]$ on all datasets, and finally we selected the mean value of the similarity matrix $\mathbf{W}$ as σ for evaluation:

$$\sigma = mean(\mathbf{W}) \tag{8}$$

Where $\mathbf{W}$ is the similarity matrix calculated by Eq. (3).

4.4 Evaluation measurements

To fully capture different aspects of the clustering result, we employed the following evaluation metrics, such as accuracy (ACC), normalised mutual information (NMI) and purity (PUR) [14]. We report the definitions of the involved evaluation metrics as below.

Accuracy (ACC) is defined as:

$$ACC = \frac{N_{cor}}{N} \tag{9}$$

Where N_{cor} denotes the number of data points falling in the correct groups.

NMI takes into account the tradeoff between quality and clusters number [12]. It is defined as:

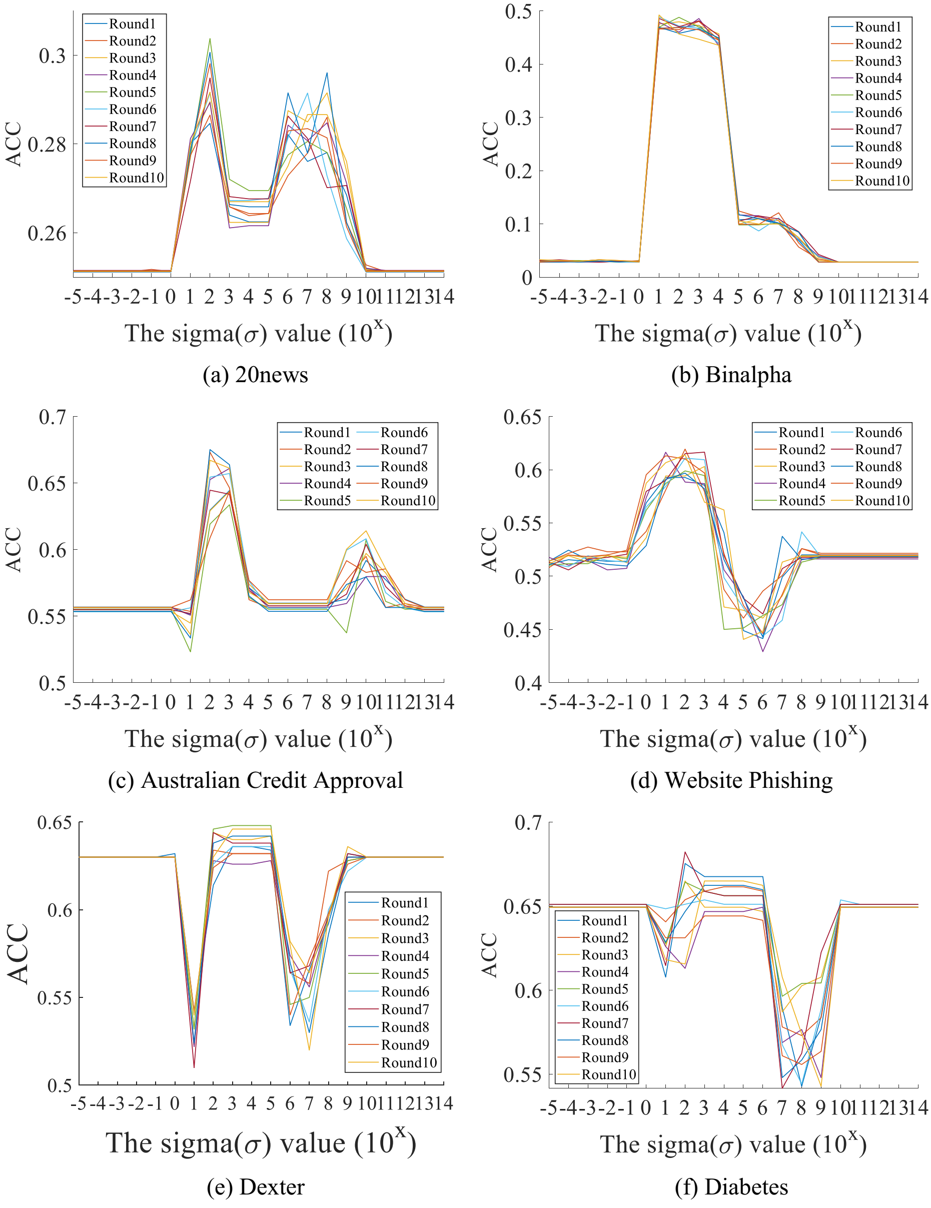
$$NMI = 2\frac{M\left(X_i, X_j\right)}{E(X_i) + E\left(, X_j\right)} \tag{10}$$

Where $M\ (X_i, X_j)$ is mutual information between two variables, and $E(\cdot)$ denotes the entropy of the variable.

PUR is used to summarise the percentage of truly classified data points in each cluster comparing to the ground truth. It is defined as:

$$PUR = \sum_{i=1}^{k} \frac{S_i}{n} P_i \tag{11}$$

Where k is clusters number and S_i is the number of data points of the i-th class. P_i denotes the distribution of correctly partitioned data points in all clusters [14].



(a) 20news

(b) Binalpha

(c) Australian Credit Approval

(d) Website Phishing

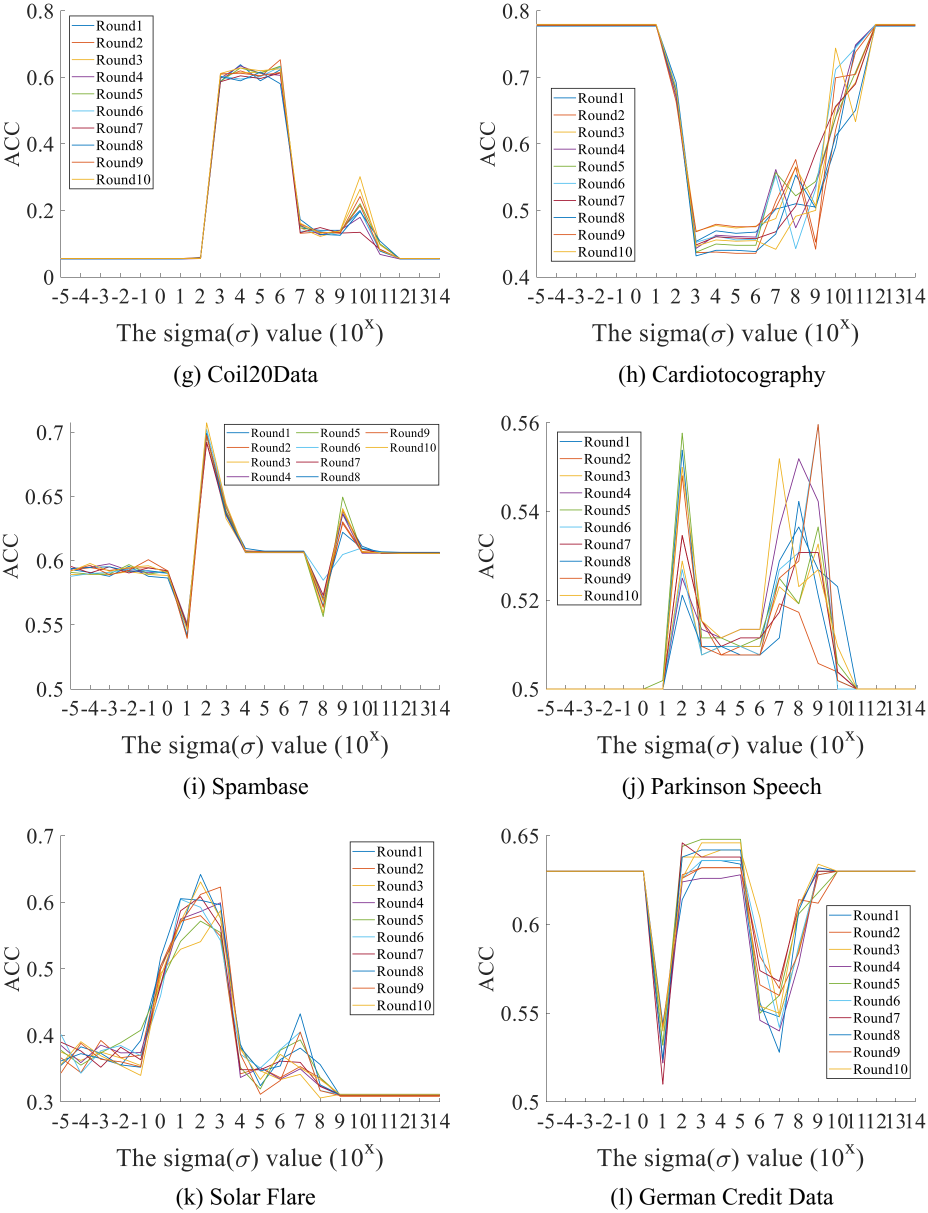
(e) Dexter

(f) Diabetes

Fig. 2 **a.** ACC trends of our WAMKM emthod with different σ values **b.** ACC trends of our WAMKM emthod with different σ values

4.5 Result analysis

Figure 2a and Fig. 2b show the trends of ACC of our proposed WAMKM method with different values of parameter σ on each dataset. Figures 3, 4 and 5 show the results of ACC, NMI and PUR in each iteration on all 12 datasets, and Fig. 6 summarises the results of Figs. 3, 4 and 5. Based on our experimental results, we have the following observations.



(g) Coil20Data

(h) Cardiotocography

(i) Spambase

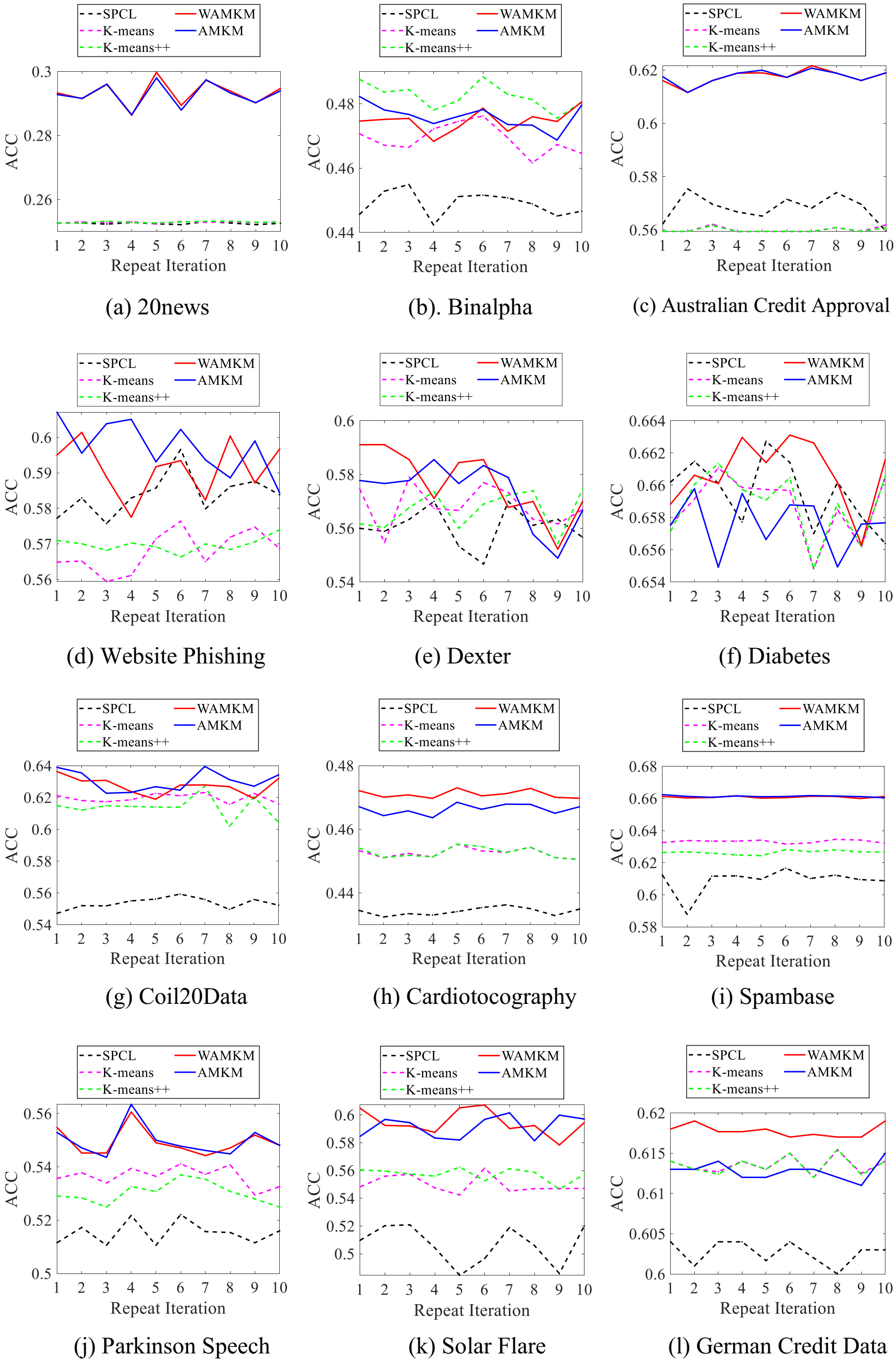
(j) Parkinson Speech

(k) Solar Flare

(l) German Credit Data

Fig. 2 (continued)

First, our proposed methods are sensitive to the setting parameter σ, which controls the similarity between two data points. For example, the ACC results first keep stable while varying the value of σ from 10^{-5} to 10^{0}, and then begin increasing gradually until arriving their peaks, i.e., 10^{10} for the value of σ on some datasets, such as 20news, Binalpha, Australian Credit Approval, Coil20Data, Parkinson Speech and Solar Flare.



(a) 20news

(b). Binalpha

(c) Australian Credit Approval

(d) Website Phishing

(e) Dexter

(f) Diabetes

(g) Coil20Data

(h) Cardiotocography

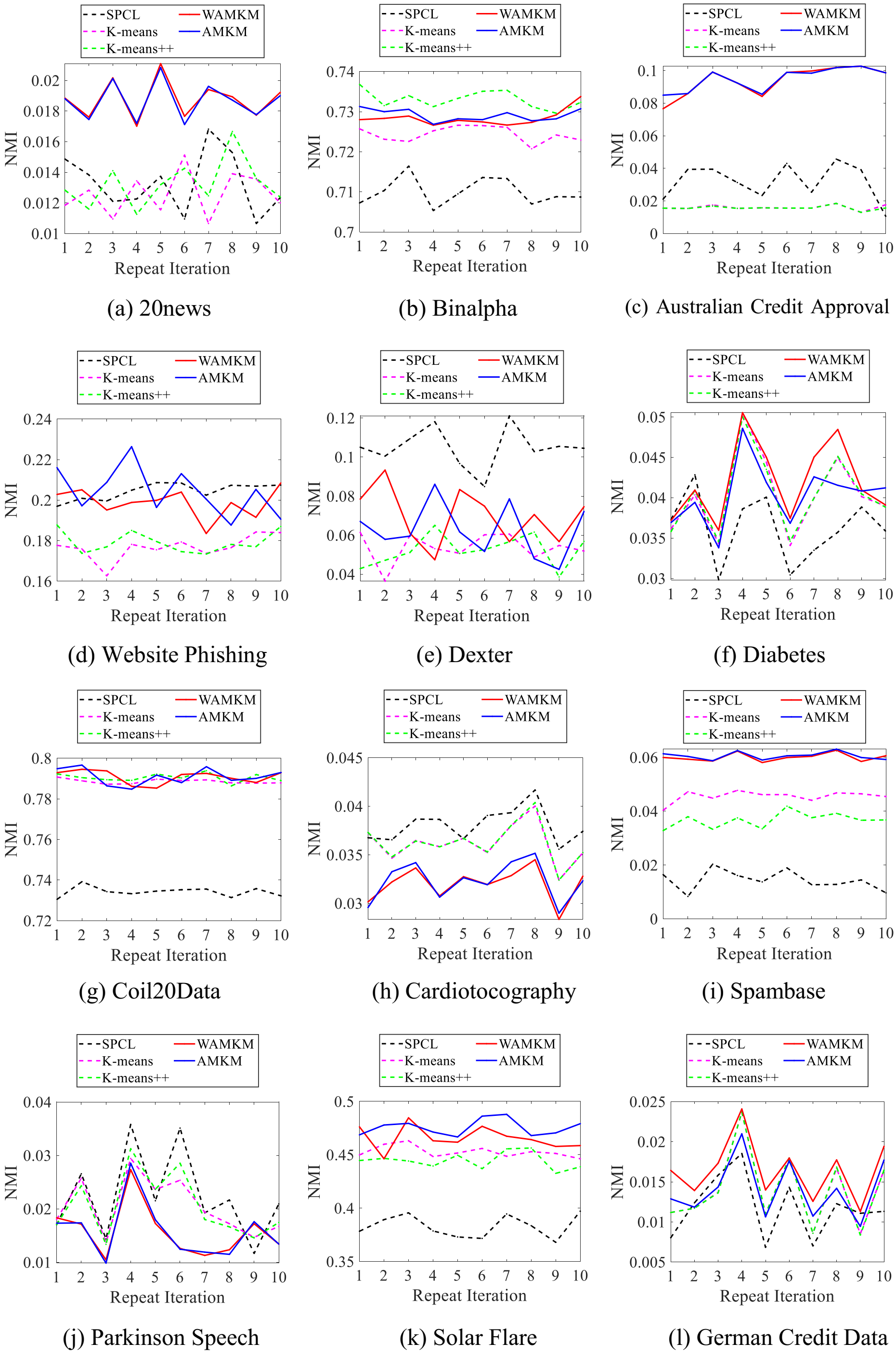
(i) Spambase

(j) Parkinson Speech

(k) Solar Flare

(l) German Credit Data

Fig. 3 ACC variations of all methods in each iteration of every dataset



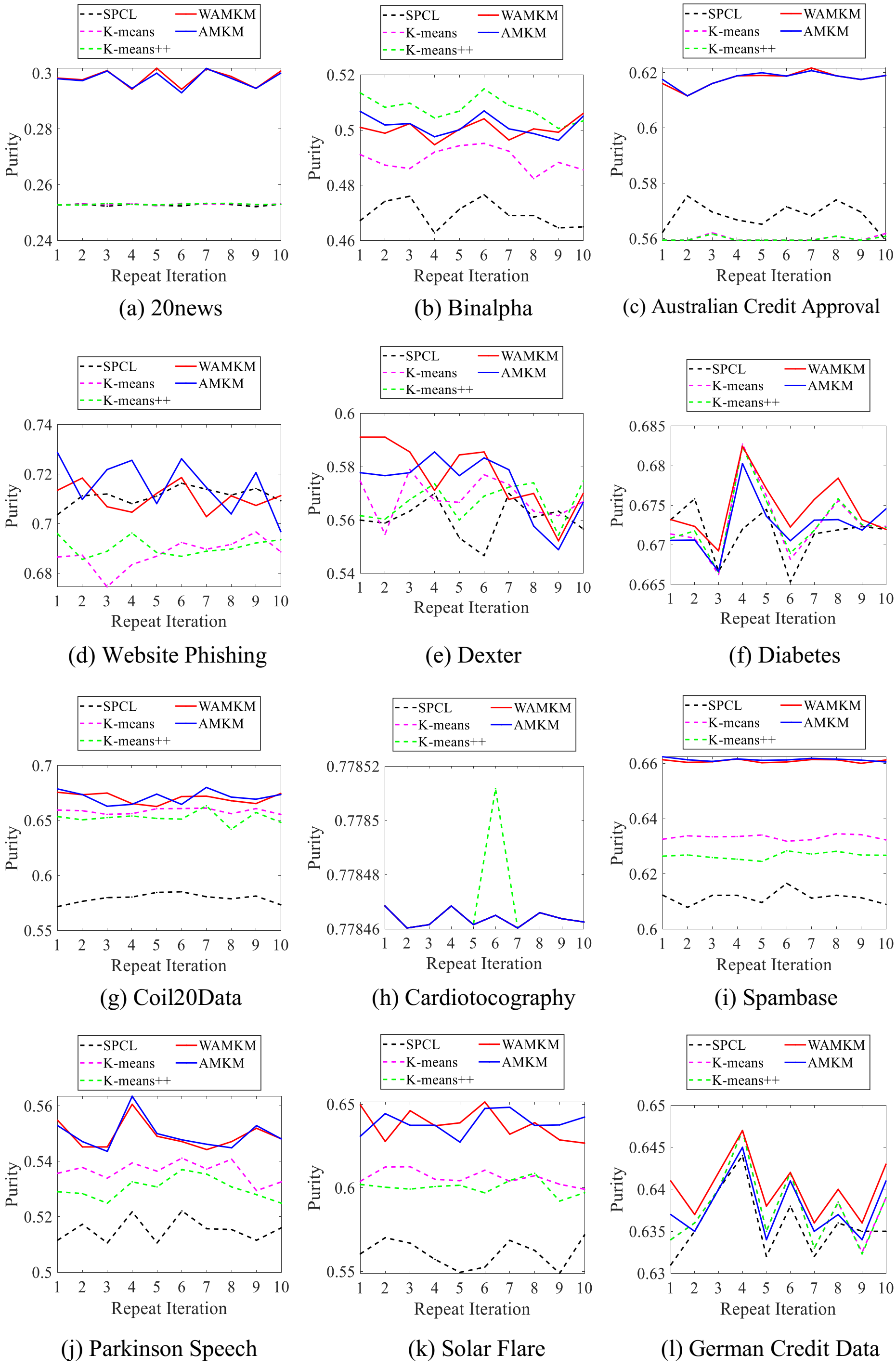
(a) 20news (b) Binalpha (c) Australian Credit Approval

(d) Website Phishing (e) Dexter (f) Diabetes

(g) Coil20Data (h) Cardiotocography (i) Spambase

(j) Parkinson Speech (k) Solar Flare (l) German Credit Data

Fig. 4 NMI variations of all methods in each iteration of every dataset



(a) 20news

(b) Binalpha

(c) Australian Credit Approval

(d) Website Phishing

(e) Dexter

(f) Diabetes

(g) Coil20Data

(h) Cardiotocography

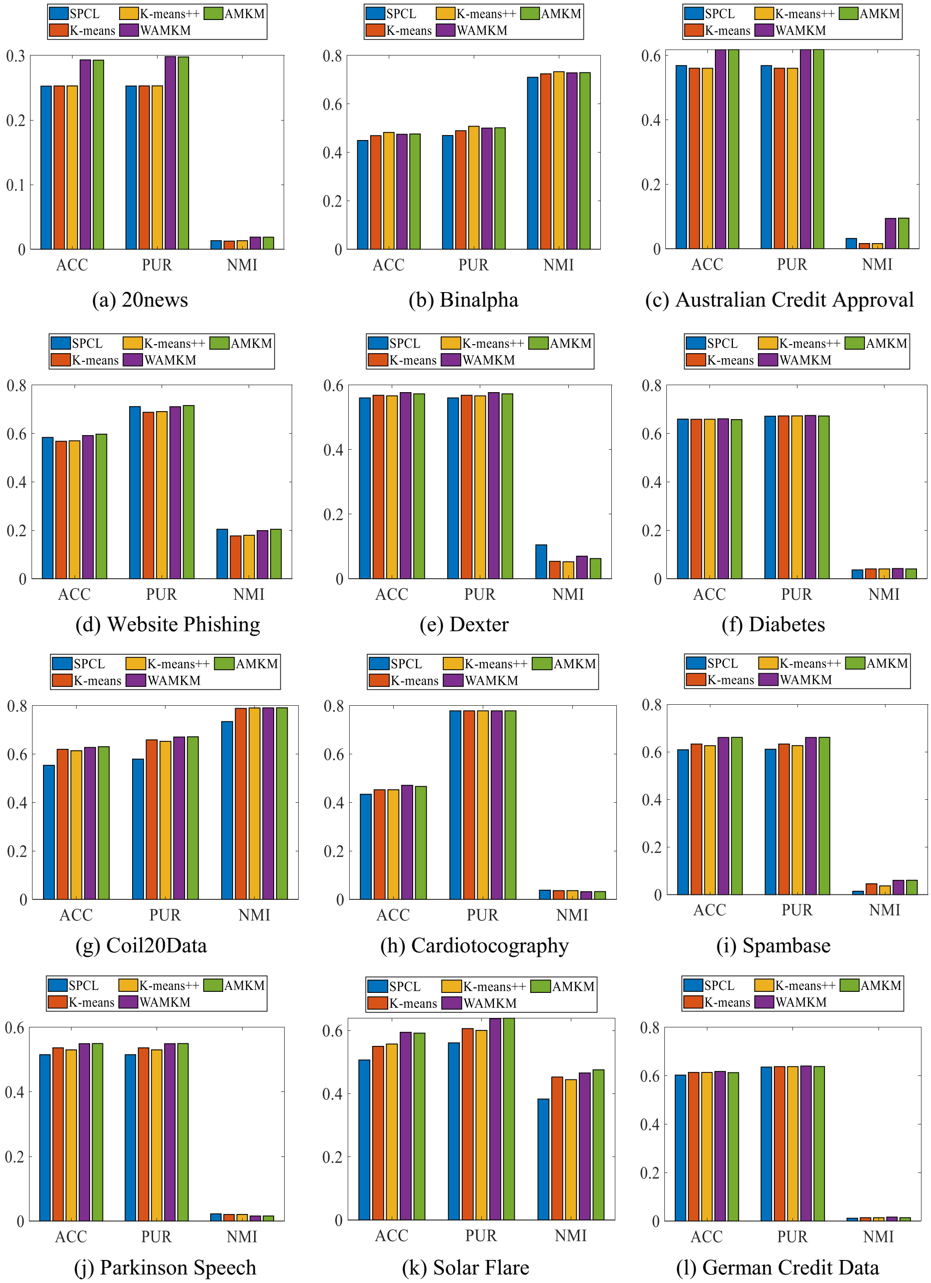
(i) Spambase

(j) Parkinson Speech

(k) Solar Flare

(l) German Credit Data

Fig. 5 PUR variations of all methods in each iteration of every dataset



(a) 20news

(b) Binalpha

(c) Australian Credit Approval

(d) Website Phishing

(e) Dexter

(f) Diabetes

(g) Coil20Data

(h) Cardiotocography

(i) Spambase

(j) Parkinson Speech

(k) Solar Flare

(l) German Credit Data

Fig. 6 The summarize results of all methods on every dataset

The ACC results of show the fluctuation trends when σ is between 10^0 and 10^{10}, and then keep stable while the value of σ is out of such a range, on other datasets, such as Website Phishing, Diabetes and Spambase. It is noteworthy that the corresponding results of our proposed AMKM have the similar trends as in Fig. 2a and b. Moreover, our proposed WAMKM is more sensitive to the setting of the parameter σ, compared our proposed method AMKM. We did not report these results due to the space limitation of this paper. In nutshell, the ACC results on our selected datasets vary while the value of parameter σ is in the range between 10^{-1} and 10^{10}. The possible reason could be that the elements of the adjacent matrix will be all nearly zero when the value of parameter σ is too small or too large. Hence, it is essential to tune the value of parameter σ carefully and accurately. Moreover, to archive the best clustering performance, different datasets should use different ranges of σ.

Second, our proposed methods outperformed the comparison methods on all datasets, in terms of three clustering evaluation metrics. For example, our proposed methods improved on average by 5.51%, 25.99%, and 3.85% respectively, compare to *k*-means, *k*-means++ and spectral clustering algorithms, in terms of ACC, NMI and PUR, on all datasets. In particular, our method achieved the most improvement by 17.4% in terms of ACC on dataset Coil20Data, 197.2% in terms of NMI on dataset Australian Credit Approval, and 17.9% in terms of PUR on dataset 20news. Furthermore, our proposed methods outperformed the comparison methods in terms of ACC, NMI, and Purity, respectively, on ten datasets, eight datasets, and nine datasets of total twelve datasets. The reason is that our proposed methods generated better representations, compared to the use of spectral representation of SPCL and the use of original features in both k-means and k-means ++. It implies that representation learning is very important for clustering analysis, which was demonstrated in the literature [43, 44].

Last but not least, our proposed WAMKM method has no significant improvements, compared to our proposed AMKM method, in terms of all three evaluation metrics. The possible reason is that the feature weight is seriously related to the quality of the similarity matrix, which is sensitive to the setting of the parameter σ. However, our proposed WAMKM method is more sensitive than our proposed method AMKM, in terms of the variations of the parameter σ.

5 Conclusions

In this paper, we have proposed two clustering methods to address the issues of previous *k*-means clustering and spectral clustering algorithms. To do this, we first devised an adjacent matrix and a weighted adjacent matrix, and then ran *k*-means clustering on those two adjacent matrices, respectively, to output the clustering results. Finally, we evaluated the clustering results against three comparison clustering algorithms, i.e., *k*-means, *k*-means++, and normalised spectral clustering algorithms, in terms of three evaluation metrics ACC, NMI and PUR. As a result, our proposed methods outperform the comparison algorithms in our experiments.

However, we found that the experiment results of our proposed methods are sensitive to parameter σ, which is used to construct the adjacent matrix. This means that our proposed clustering methods are data-driven and their performance varies on different types of datasets. In our future work, we will extend our research to dynamically select suitable parameters.

References

1. Abe S (2010) Feature selection and extraction, in support vector machines for pattern classification. Springer, London, pp 331–341
2. Arora P, Varshney S (2016) Analysis of k-means and k-medoids algorithm for big data. Proc Comput Sci 78:507–512
3. Bachem O et al. (2016) Approximate K-means++ in sublinear time. AAAI. Phoenix, Arizona USA: 1459–1467
4. Birch ZT (1996) BIRCH: an efficient data clustering method for very large databases, in SIGMOD, R.R. T. Zhang, M. Livny, Editor. New York: 103–114
5. Bryant A, Cios K (2018) RNN-DBSCAN: a density-based clustering algorithm using reverse nearest neighbor density estimates. IEEE Trans Knowl Data Eng 30(6):1109–1121
6. Capó M, Pérez A, Lozano JA (2017) An efficient approximation to the K-means clustering for massive data. Knowl-Based Syst 117:56–69
7. Cassisi C et al (2013) Enhancing density-based clustering: parameter reduction and outlier detection. Inf Syst 38(3):317–330
8. Chang L et al (2017) Fast and exact structural graph clustering. IEEE Trans Knowl Data Eng 29(2):387–401
9. Chen J et al (2018) FGCH: a fast and grid based clustering algorithm for hybrid data stream. Appl Intell 49(4):1228–1244
10. Deng Z et al (2015) A scalable and fast OPTICS for clustering trajectory big data. Clust Comput 18(2):549–562
11. Ding Y, Fu X (2016) Kernel-based fuzzy c-means clustering algorithm based on genetic algorithm. Neurocomputing 188:233–238
12. Domeniconi C, Al-Razgan M (2009) Weighted cluster ensembles: methods and analysis. ACM Trans Knowledge Discov Data (TKDD) 2(4):17–57
13. Du L et al. (2015) Robust multiple kernel K-means using L21-norm. IJCAI.: Buenos Aires, Argentina: 3476–3482
14. Du, T., et al., (2018) Spectral clustering algorithm combining local covariance matrix with normalization. Neural Comput & Applic: 1–8.
15. Ferreira, M.R.P., F.d.A.T. de Carvalho, and E.C. Simões, Kernel-based hard clustering methods with kernelization of the metric and automatic weighting of the variables. Pattern Recogn, 2016. 51: p. 310–321.
16. Gan J, Tao Y (2015) DBSCAN revisited: mis-claim, un-fixability, and approximation. SIGMOD: Melbourne: 519–530
17. Gebru ID et al (2016) EM algorithms for weighted-data clustering with application to audio-visual scene analysis. IEEE Trans Pattern Anal Mach Intell 38(12):2402–2415
18. Guha S, Rastogi R, Shim K (1998) CURE: an efficient clustering algorithm for large databases, in SIGMOD. ACM, Seattle, Washington, USA, pp 73–84
19. Guha S, Rastogi R, Shim K (2000) ROCK: a robust clustering algorithm for categorical attributes. Inf Syst 25(5):345–366
20. He L et al (2018) Fast large-scale spectral clustering via explicit feature mapping. IEEE Transactions on Cybernetics 49(3):1058–1071
21. Jayasumana S et al (2015) Kernel methods on Riemannian manifolds with Gaussian RBF kernels. IEEE Trans Pattern Anal Mach Intell 37(12):2464–2477
22. Kodinariya TM, Makwana PR (2013) Review on determining number of cluster in K-means clustering. Int J 1(6):90–95
23. Lattanzi S et al (2015) Robust hierarchical k-Center clustering, in ITCS. ACM, Rehovot, Israel, pp 211–218
24. Lei C, Zhu X (2018) Unsupervised feature selection via local structure learning and sparse learning. Multimed Tools Appl 77(22):29605–29622
25. Liu F, Ye C, Zhu E (2017) Accurate grid-based clustering algorithm with diagonal grid searching and merging. ICAMMT. https://doi.org/10.1088/1757-899X/242/1/012123
26. Lv Y et al (2016) An efficient and scalable density-based clustering algorithm for datasets with complex structures. Neurocomputing 171:9–22
27. Malinen MI, Fränti P (2014) Balanced K-means for clustering. S+SSPR. : Berlin, Heidelberg: 32–41
28. Murtagh F, Contreras P (2012) Algorithms for hierarchical clustering: an overview. Wiley Data Mining Knowl Discov 2(1):86–97
29. Ng AY, Jordan MI, Weiss Y (2002) On spectral clustering: Analysis and an algorithm, in NIPS: 849–856
30. Pavan KK, Rao AD, Sridhar G (2010) Single pass seed selection algorithm for k-means. Comput Sci 6(1): 60–66
31. Sharma A, Sharma A (2017) KNN-DBSCAN: Using k-nearest neighbor information for parameter-free density based clustering, in ICICICT: Kannur, India: 787–792

32. Shiokawa H, Fujiwara Y, Onizuka M (2015) SCAN++: efficient algorithm for finding clusters, hubs and outliers on large-scale graphs. Proc VLDB Endow 8(11):1178–1189
33. Shiokawa H, Takahashi T, Kitagawa H (2018) ScaleSCAN: scalable density-based graph clustering, in *DEXA*: Cham: 18–34
34. Souza CR (2010) Kernel functions for machine learning applications. Creative Commons Attribution-Noncommercial-Share Alike 3:29–41
35. Sting WWYJMR (1997) A Statistical Information Grid Approach to Spatial Data Mining, in VLDB. : Athens, Greece: 186–195
36. Tibshirani R, Walther G, Hastie T (2002) Estimating the number of clusters in a data set via the gap statistic. J Royal Stat Soc: Ser B (Statistical Methodology) 63(2):411–423
37. Tremblay N et al. (2016) Compressive spectral clustering. ICML : New York: 1002–1011
38. Vajda S, Santosh KC (2016) A fast k-nearest neighbor classifier using unsupervised clustering, in *RTIP2R* : Singapore: 185–193
39. Von Luxburg U (2007) A tutorial on spectral clustering. Stat Comput 17(4):395–416
40. Xu D, Tian Y (2015) A comprehensive survey of clustering algorithms. Ann Data Sci 2(2):165–193
41. Xu X et al. (2007) Scan: a structural clustering algorithm for networks. In KDD. ACM: San Jose, CA: 824–833
42. Zahra S et al (2015) Novel centroid selection approaches for KMeans-clustering based recommender systems. Inf Sci 320:156–189
43. Zhang S (2018) Multiple-scale cost sensitive decision tree learning. World Wide Web 21:1787–1800. https://doi.org/10.1007/s11280-018-0619-5
44. Zhang S (2019) Cost-sensitive KNN classification. Neurocomputing. https://doi.org/10.1016/j.neucom.2018.11.101
45. Zheng W et al (2017) Dynamic graph learning for spectral feature selection. Multimed Tools Appl 77(22): 29739–29755
46. Zheng W et al (2018) Unsupervised feature selection by self-paced learning regularization. Pattern Recogn Lett. https://doi.org/10.1016/j.patrec.2018.06.029
47. Zhu X, Li X, Zhang S (2015) Block-row sparse multiview multilabel learning for image classification. IEEE Trans Cybernet 46(2):450–461
48. Zhu X et al (2017) Graph PCA hashing for similarity search. IEEE Trans Multimed 19(9):2033–2044
49. Zhu X et al (2018) Low-rank sparse subspace for spectral clustering. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2018.2858782
50. Zhu X et al (2018) One-step multi-view spectral clustering. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2018.2873378

Jukai Zhou is a Master student at SNCS of Massey University, New Zealand. His research interests include data mining, machine learning and Big Data computing.

Tong Liu is a faculty member at the School of Natural and Computational Sciences, Massey University, Auckland, New Zealand. She holds a Master of Science degree in Computer Science from Massey University, New Zealand. Her research interest includes big data, data mining, machine learning, artificial intelligence, software engineering, and application of IT in industry.

Jinting Zhu is a Ph.D. candidate at the School of Natural and Computational Sciences, Massey University, New Zealand. He holds a Master's degree in Computer Science from Kunming University of Science and Technology, China. His research interest includes data analysis, and multimedia application.