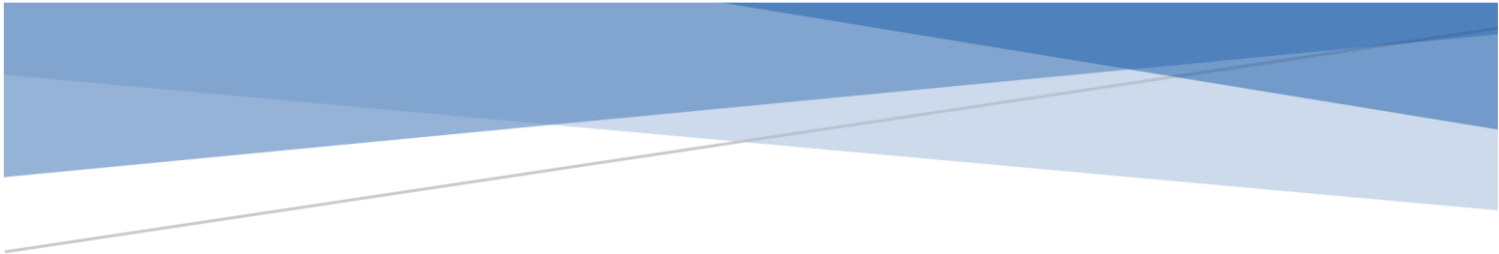


Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.



# **RESEARCH ON ADJACENT MATRIX FOR *K*-MEANS CLUSTERING**

A thesis presented for the degree of  
Master of Computer Science

In

School of Natural and Computational Sciences  
At Massey University, Auckland  
New Zealand

By

Jukai Zhou (z.jukai@massey.ac.nz)

June 2019

## Acknowledgement

Firstly, I would express my genuine gratitude to my supervisor Prof. Sean Zhu for his continuous support during the entire duration of my study and research of this Master degree. His immense knowledge, patience and enthusiasm helped me across all the aspects of my research and study, as well as he guided me throughout my thesis for the numerous consultations of presentation and layout.

Besides my supervisor, I would like to say thanks to my manager and my colleagues as well as the head of school, for their full understanding and support, in both workload arrangement and funding of finance. I would also like to expand my gratitude to all people who have helped me in composing this thesis directly and indirectly.

Last but not the least, I would like to say thanks to my parents and my family for their full supporting me in studying this Master qualification. Without their encouragements, it is not possible to accomplish my study in time.

## Abstract

Machine learning is playing a vital role in our modern world. Depending on whether the data has labels or not, machine learning mainly contains three categories, i.e., unsupervised learning, supervised learning, and semi-supervised learning. As labels are usually difficult and expensive to be obtained, unsupervised learning is more popular, compared to supervised learning and semi-supervised learning. Moreover,  $k$ -means clustering is very popular in the domain of unsupervised learning. Hence, this thesis focuses on the improvement of previous  $k$ -means clustering.

$K$ -means clustering has been widely applied in real applications 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. Hence,  $k$ -means clustering is still a hot research topic in unsupervised learning. In this thesis, we propose to improve traditional  $k$ -means clustering by designing two different similarity matrices to represent the original data points.

The first method first constructs a new representation (i.e., an adjacent matrix) to replace the original representation of data points, and then runs  $k$ -means clustering on the resulted adjacent matrix. In this way, our proposed method benefits from the high-order similarity among data points to capture the complex structure inherent in data points as well as avoids the time-consuming process of eigenvectors decomposition in spectral clustering.

The second method takes into account the weights of the features to improve the former method, based on the assumption that different features contain different contributions to the construction of the clustering models. As a result, it makes the clustering model more robust, compared to the first method as well as previous clustering methods.

Finally, we tested our proposed clustering methods on public UCI datasets. Experimental results showed the clustering results of our proposed methods significantly outperformed the comparison methods in terms of three evaluation metrics.

**Keywords:**  $k$ -means clustering, similarity measurement, adjacent matrix, unsupervised learning.

## Table of Contents

Acknowledgement.....	i
Abstract.....	ii
Chapter. 1. Introduction.....	1
1.1 Background.....	1
1.2 Unsupervised learning .....	4
1.3 Research motivation.....	5
1.4 Summary.....	7
Chapter. 2. Literature Review .....	8
2.1 Partition based clustering algorithms .....	8
2.2 Hierarchy based clustering algorithms.....	8
2.3 Density based clustering algorithms .....	9
2.4 Graph based clustering algorithms.....	10
2.5 Grid based clustering algorithms .....	11
2.6 Kernel based algorithms .....	12
2.7 Summary of clustering algorithm .....	12
Chapter. 3. Preliminary and Motivation.....	13
3.1 Notations.....	13
3.2 $K$ -means clustering .....	13
3.3 $K$ -means clustering issues .....	16
3.4 Spectral clustering introduction .....	22
3.5 Summary of $k$ -means clustering and SPCL .....	29
Chapter. 4. Proposed Methods .....	30
4.1 AMKM .....	31
4.2 Weighted AMKM.....	34
4.3 Summary of AMKM and WAMKM .....	37
Chapter. 5. Experimental Analysis.....	38
5.1 Datasets.....	38
5.2 Comparison algorithm .....	40
5.3 Evaluation Method.....	41

5.4	Parameter Setting .....	42
5.5	Evaluation Measurement .....	42
5.6	Experiment Result.....	43
5.7	Result analysis .....	52
Chapter. 6. Conclusion and Future Work.....		54
6.1	Conclusion .....	54
6.2	Future Work.....	54
Appendices .....		55
Appendix A.	Matlab code for AMKM .....	55
Appendix B.	Matlab code for WAMKM .....	56
Appendix C.	Matlab code for $k$ -means clustering .....	57
Appendix D.	Matlab code for performance evaluation .....	59
References .....		62

## Chapter. 1. Introduction

### 1.1 Background

With the development of information technology, the society generates a huge number of data. Normally, it is essential to obtain knowledge from the data for serving the society. However, manually mining useful knowledge from massive data is usually time-consuming and difficult as the number of the data is massive and the structure of the data is complex. Machine learning is a good alternative to address this issue as it could enable computers automatically obtaining knowledge by exploring the structure inherent in the data, and has been becoming increasingly popular in our real life. More specifically, machine learning usually uses two steps to mine knowledge from the data, i.e., the training process and the testing process. In the training process, machine learning methods are designed to handle the training data for outputting a model, while the result model is used to conduct prediction in the testing process. Based on different criteria, different machine learning methods satisfy various requirements. The popular machine learning methods include anomaly detection, dimensionality reduction, clustering, classification, regression, and so on. Depending on whether the training data is labelled or not, existing machine learning methods are typically divided into three categories, i.e., unsupervised learning, supervised learning, and semi-supervised learning [1]. We list the category of machine learning methods in Fig. 1 and introduce the details as follows.

Machine Learning						
Unsupervised Learning			Supervised Learning		Semi-supervised Learning	
Clustering	Dimensionality reduction	Anomaly Detection	Classification	Regression	Transductive	Inductive
<ul style="list-style-type: none"> <li>• Partition based algorithms</li> <li>• Graph base algorithms</li> <li>• Hierarchy based algorithms</li> <li>• Density based algorithms</li> <li>• Kernel based algorithms</li> <li>• Grid based algorithms</li> <li>...</li> </ul>	<ul style="list-style-type: none"> <li>• Principal component analysis (PCA)</li> <li>• Non-negative matrix factorization (NMF)</li> <li>• Graph-based kernel PCA</li> <li>• Linear discriminant analysis (LDA)</li> <li>• Generalised discriminant analysis (GDA)</li> <li>• Kernel PCA</li> <li>...</li> </ul>	<ul style="list-style-type: none"> <li>• Density-based techniques</li> <li>• Replicator neural networks</li> <li>• Bayesian Networks</li> <li>• Hidden Markov models (HMMs)</li> <li>• Fuzzy logic-based techniques</li> <li>• Ensemble techniques</li> <li>...</li> </ul>	<ul style="list-style-type: none"> <li>• Linear classifiers</li> <li>• Support vector machine (SVM)</li> <li>• Quadratic classifiers</li> <li>• Kernel estimation</li> <li>• Boosting</li> <li>• Decision tree</li> <li>• Neural networks</li> <li>• Learning vector quantization</li> <li>...</li> </ul>	<ul style="list-style-type: none"> <li>• Linear regression</li> <li>• Logistic regression</li> <li>• Polynomial regression</li> <li>• Stepwise regression</li> <li>• Elastic-net regression</li> <li>• Ridge regression</li> <li>• Lasso regression</li> <li>...</li> </ul>	<ul style="list-style-type: none"> <li>• Generative models</li> <li>• Heuristic methods</li> <li>• Mixture models</li> <li>• Multi-view learning</li> <li>• Yarowsky algorithm</li> <li>• Transductive SVM</li> <li>• Concluding thoughts</li> <li>...</li> </ul>	<ul style="list-style-type: none"> <li>• Low-density separation</li> <li>• Graph-based methods</li> <li>• Self-training</li> <li>• Co-training</li> <li>• Entropy regularisation</li> <li>• Cluster-and-Label Approach</li> <li>• Expectation Maximisation Approach</li> <li>...</li> </ul>

**Fig. 1.** The category of machine learning methods.

Unsupervised learning models the underlying structure or distribution of the data without the need of labels [2]. It allows to deal with the data without knowing its internal relationship, aim at automatically outputting the structure/pattern of the data. Unsupervised learning has been applied in all kinds of applications, such as motion, image and voice recognition, search engine and online security, social media and health care, financial and marketing services, and so on. The typical methods of unsupervised learning include anomaly detection, dimensionality reduction, clustering, and so on.

Supervised learning captures the relationship between input features of the data and their corresponding labels [3]. Here we let  $\mathbf{X}$  and  $\mathbf{Y}$  denote the data and label matrices,  $n$ ,  $d$  and  $c$  denote the matrix dimensions, respectively. In the training process, given a feature training matrix  $\mathbf{X} \in \mathbf{R}^{n \times d}$  where each row represents a sample or a data point, and the corresponding decision matrix  $\mathbf{Y} \in \mathbf{R}^{n \times c}$ , supervised learning tries to learn the relationship between  $\mathbf{X}$  and  $\mathbf{Y}$ , i.e.,  $\mathbf{Y} = f(\mathbf{X})$ . In the testing process, the



learnt relationship  $f(\mathbf{X})$  can be used to obtain the prediction  $f(\mathbf{X}_t)$  for the testing matrix  $\mathbf{X}_t$ . The most popular methods of supervised learning include regression methods (i.e., every element of  $\mathbf{Y}$  is a continuous value) and classification methods (i.e., every element of  $\mathbf{Y}$  is a discrete value). The typical classification methods include linear classifiers, support vector machine (SVM) [4], quadratic classifiers, kernel estimation, boosting, decision tree [5], neural networks, learning vector quantization, etc. The typical regression models include ridge regression, linear regression, lasso regression, elastic-net regression, logistic regression, polynomial regression and stepwise regression, etc. [6].

Semi-supervised learning is a special case that combines the techniques of both unsupervised learning and supervised learning, with a large amount of unlabeled data and less labelled data for the training process [7]. Supervised learning needs enough labelled data to construct robust models. However, labels are usually not easy to be collected due to all kinds of reasons, such as scarce labelled data, time-consuming to obtain labelled data. Therefore, semi-supervised learning is able to improve the limitation of supervised learning on limited labelled data by using available unlabelled data as well as labelled data. The typical semi-supervised learning algorithms include graph-based methods, generative models, self-training, heuristic methods, low-density separation, mixture models, co-training and multi-view learning [8], etc.

In a word, all the three types of machine learning methods have widely been applied in real applications. However, since unlabelled data

is relatively easy to obtain, thus unsupervised learning has been developing increasingly rapid and has attracted more interests than the other two [9]. Hence, this thesis focuses on unsupervised learning.

## **1.2 Unsupervised learning**

In literature, the popular methods of unsupervised learning include anomaly detection, dimensionality reduction, clustering, and so on.

### **1.2.1 Anomaly detection**

Anomaly detection (also called outlier detection) is to identify rare items, behaviors or observations that differ from the majority of the data significantly [10]. It is usually used to raise suspicions or to find the outliers in a dataset. The typical algorithms include density-based techniques, replicator neural networks, fuzzy logic-based techniques, Bayesian networks, cluster analysis-based techniques, hidden Markov models (HMMs), ensemble techniques [11], etc.

### **1.2.2 Dimensionality reduction**

Dimensionality reduction is to reduce the feature number of high-dimensional data. It is a commonly and widely used technique to improve the efficiency of machine learning without losing too much accuracy by removing some unimportant features [12]. After conducting dimensionality reduction, machine learning algorithms are able to work on large datasets with high efficiency and effectiveness. The typical algorithms of dimensionality reduction include principal component analysis (PCA),

kernel PCA, non-negative matrix factorization (NMF), graph-based kernel PCA, generalised discriminant analysis (GDA) and linear discriminant analysis (LDA) [13], etc.

### 1.2.3 Clustering

Clustering is to divide a set of data points into groups, where similar data points are in the same groups and dissimilar data points are in different groups [13]. Specifically, clustering computes the similarities of features among all of the data points, and then group them together by similarities. Clustering is very useful in some situations, e.g., decision-making, image processing and pattern analysing. The typical clustering techniques include partition based algorithms, grid based algorithms, hierarchy based algorithms, density based algorithms, graph based algorithms and kernel based algorithms, etc.

Clustering is a vital part of unsupervised learning, and has been applied in many kinds of applications, such as image compression and segmentation, document classification, Cyber security and fraud proofing, and so on. In this thesis, we focus on overcoming the drawbacks of previous clustering methods.

## 1.3 Research motivation

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 [14]. In the past years much

efforts have been devoted for addressing these issues, such as rule of thumb method [15] and gap statistic method [16] for selecting the optimal value of  $k$ , hierarchical centroid selection and simple cluster seeking [17] for centroid initialisation, self-paced learning technique [18] and multiple feature extraction algorithm [19] for constructing the similarity matrix. Another popular clustering algorithm is spectral clustering, which uses spectral representation (measuring the relationship among data points, as known as the high-order relationship [20]) 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 [21, 22].

Based on the observations above, in this thesis, we focus on investigating an effective similarity matrix for addressing the third limitation of  $k$ -means clustering, i.e., the construction of an efficient similarity matrix [23]. With the help of the efficient similarity matrix, our proposed methods improved 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. Comparing with traditional  $k$ -means clustering, our methods use the high-order similarities instead the original data points. Comparing

with spectral clustering, our clustering methods avoid both decomposition of eigenvectors and dimensionality reduction. Thus, our methods are able to not only improve the clustering accuracy but also reduce the computational complexity as well.

#### **1.4 Summary**

The rest of this thesis is organised as below. In Chapter 2, we conduct a comprehensive literature review of the current clustering techniques, and then introduce two important clustering algorithms, i.e.,  $k$ -means clustering and spectral clustering in Chapter 3. We propose our clustering methods in Chapter 4, followed by a detailed demonstration. In Chapter 5, we conduct experiments on real UCI datasets and results analysis by comparing our proposed clustering methods with  $k$ -means,  $k$ -means++ and spectral clustering algorithms, in terms of the evaluation metrics. Finally, in Chapter 6, we conclude and summarise our work, followed by some future research work. We also attach our referred journals and books, and implementing code in the parts of references and appendices, respectively.

## Chapter. 2. Literature Review

In the literature, clustering algorithms are partitioned into the following categories, such as partition based clustering algorithms, hierarchy based clustering algorithms, density based clustering algorithms, grid based clustering algorithms, kernel based clustering algorithms and graph based clustering algorithms. We introduce them in details as bellows.

### 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 [24] and  $k$ -means++ [25]. Recently, both balanced  $k$ -means [26] and recursive partition based  $k$ -means [27] 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 hierarchy based clustering algorithms 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 systematically 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) [28] , clustering using representatives (CURE) [29] and robust clustering using links (ROCK) [30] . However, most hierarchical clustering algorithms are sensitive to noise, indicating that the clustering result may be affected by even few minor outliers [31]. Hence, some enhanced hierarchical clustering algorithms, e.g., robust hierarchical  $k$ -center clustering [32], are developed to address this issue.

### **2.3 Density based clustering algorithms**

The most important principle of density based clustering algorithms is on the assumption 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) [33] and ordering points to identify

the clustering structure (OPTICS) [34] are the conventional representatives of density based clustering algorithms, while influence space DBSCAN [35] and DBSCAN based on influence space and detecting of border points [36] are their revised versions. Most recently, RNN-DBSCAN [37] uses the number of reverse nearest neighbours as an estimate of observation density, while  $k$ -nearest neighbor DBSCAN [38] 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 [39] [40]. 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) [41] is a classic representative of graph based clustering algorithms which aims to find out the minimum



weight division of the graph iteratively. Other graph based clustering algorithms include structural clustering algorithm for networks (SCAN) [42], SCAN++ [43], pruned SCAN (pSCAN) [44] and Scalable Density-Based Graph Clustering (ScaleSCAN) [45].

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 [46], fast large-scale spectral clustering via explicit feature mapping [47], and one-step multi-view spectral clustering [21].

## **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 grid (STING) [48] takes advantage of both grid clustering algorithm and parallel computing. Recently, a novel grid based clustering algorithm for hybrid data stream (FGCH) [49] is designed for dealing with hybrid data, and another improved grid-based clustering algorithm called DSM in [50], is a revised version of traditional grid-based clustering algorithm that incorporates the technique of 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 [51] 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 [52] and the robust multiple kernel  $k$ -means clustering [53] have been proven to be able to improve clustering performance significantly by using kernel theory.

## **2.7 Summary of clustering algorithm**

In this chapter, we discussed some common and widely used algorithms of unsupervised learning, i.e., clustering. We discussed their theories, implementations, followed by comparing their advantages and disadvantages. We also listed their typical applications and representative algorithms, respectively.

## Chapter. 3. Preliminary and Motivation

### 3.1 Notations

In this thesis, we denote matrices, vectors, and scalars, respectively, as boldface capital letters, boldface lowercase letters, and italic letters. We summarize other notations used in this paper in Table 1.

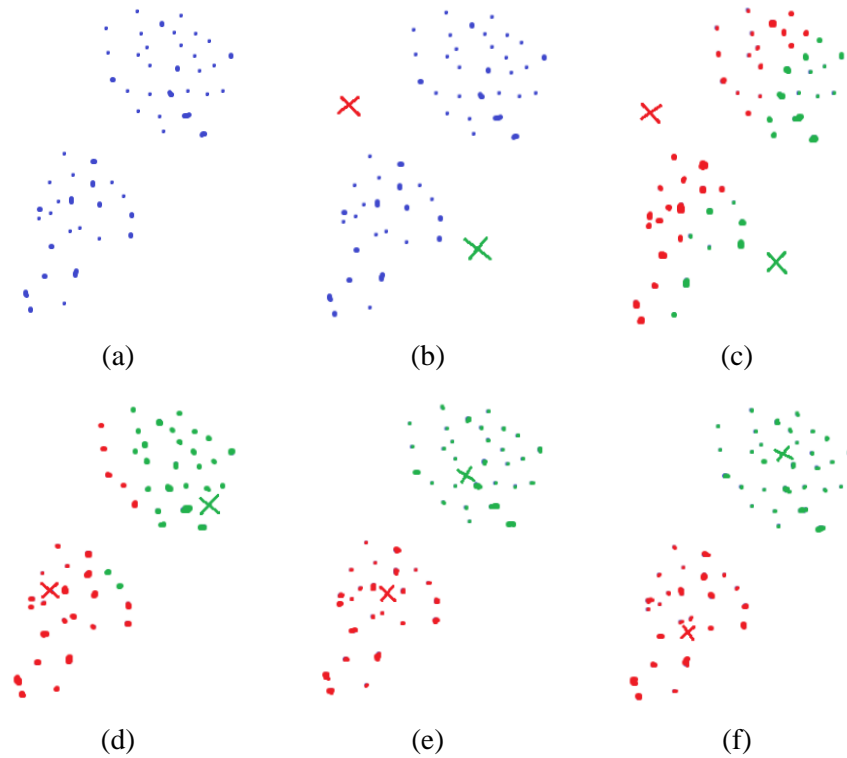
**Table 1.** Notations used in this paper

Variables	Description
$\mathbf{X}$	Matrix $\mathbf{X}$
$\mathbf{x}_i$	The $i$ -th row of $\mathbf{X}$
$\mathbf{x}^j$	The $j$ -th column of $\mathbf{X}$
$x_{i,j}$	The element of $i$ -th row and $j$ -th column in matrix $\mathbf{X}$
$\ \mathbf{x}_i - \mathbf{x}_j\ _2$	The $l_2$ norm of $\mathbf{x}_i - \mathbf{x}_j$

### 3.2 $K$ -means clustering

In the past decades,  $k$ -means clustering algorithm and its variants are the most popular partitioning algorithms and have been widely using in the machine learning fields, e.g., vector quantization, signal processing, data mining, and so on. Some revised versions also have emerged in recent years with enormous improvements and amendments to address the shortcomings of the old versions.  $K$ -means clustering algorithm and its variants are also combined with other types of clustering algorithms such as kernel based clustering algorithms, density based algorithms and

graph based clustering algorithms, etc., in order to improve the clustering performance and efficiency.



**Fig. 2.** Illustration of  $k$ -means clustering algorithm. Dots denote data points and crosses denote centroids. (a) Original data points. (b) Randomly initialised centroids. (c-f) Process of two clustering iterations. In every iteration, each data points are allocated to the closest centroid (the data point is shown as the same colour as the centroid to which is allocated), and then each cluster centroid is replaced by the mean of the data points belong to it.

Generally speaking,  $k$ -means clustering is designed to separate a group 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. To do this,  $k$ -means clustering firstly selects  $k$  number of data points at random from the dataset as the

centres of each cluster, termed as centroids, and then computes the distances between every data point and the  $k$  centroids, respectively. Secondly,  $k$ -means clustering assigns each data point to the cluster whose centroid has the closest distance to this data point to output the initial  $k$  clusters. Thirdly,  $k$ -means clustering calculates the mean value of all the data points within each cluster and update centroids for the corresponding cluster. This procedure recurs iteratively until the centroids converges and the no longer change, thus the  $k$  cluster labels and centroids formed.

We list the detailed illustration of the implementing process of  $k$ -means clustering algorithm in Fig. 2, and we also list the pseudo code of  $k$ -means clustering in Table 2 below.

**Table 2.** The pseudo code of  $k$ -means clustering

---

<b>Input:</b>	data points $\mathbf{X}=\{ \mathbf{x}_1, \dots, \mathbf{x}_n \} \in \mathbb{R}^{n \times d}$ ; the cluster number $k$ .
<b>Output:</b>	the cluster indicators of all data points and centroids $\mathbf{C}$ .
<b>1:</b>	Centroid initialisation by randomly selecting $k$ data points;
<b>2: do</b>	
<b>3:</b>	Assign data points to the closest centroids to form $k$ clusters;
<b>4:</b>	Update each centroid by the mean value of data points within each cluster;
<b>5: until</b>	
<b>6:</b>	Algorithm converges and centroids have no changes.

---

Actually, the goal of  $k$ -means clustering is to achieve the minimal sum-squared-error (SSE), which means the minimal total intra-cluster variance by a given  $k$ . The Eq. (1) below is the definition of SSE.

$$\text{SSE} = \sum_{j=1}^k \sum_{i=1}^{t_j} \|\mathbf{x}_i - \mathbf{c}_j\|_2^2 \quad (1)$$

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 clusters number and defining the similarity measurements are also major issues of  $k$ -means clustering. We introduce these issues as follows.

### 3.3 $K$ -means clustering issues

As  $k$ -means clustering and its variants are easy to implement and the performance is usually acceptable so they are widely used in different fields. Under most circumstances, the performance of  $k$ -means clustering is relatively good for most global shape clusters with numerical datasets. However, there three major problems for  $k$ -means clustering as below:

- How to identify the number of clusters, i.e., the initial value of  $k$ ?
- How to identify the optimal initial centroids?
- How to define the best similarity (or distance) measurements?

These problems above sometime pose significant influences on the clustering results and performance. We introduce those problems and their corresponding resolutions in details as bellows.

### **3.3.1 The issue of setting the value of $k$**

In real applications, the actual number of clusters, i.e., the value of  $k$ , is always unknown and there is no efficient solution in theory to identify it, so a number of literatures are targeting on solving this issue. For example, on-demand selection algorithm manually selects the value of  $k$  as the actual clusters 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 a simple equation to obtain the value of  $k$ . Below is a brief introduction of these methods above.

#### **1) On-demand selection method**

The on-demand selection method manually selects the value of  $k$  to run  $k$ -means clustering. For some datasets whose dimensions are low and visible, the value of  $k$  can be identified manually by requirements, or by the already known groups. For example, people can be partitioned by age, gender and citizenship, while shopping stores can be partitioned by revenue, type of products, or geographical locations. These attributes are always in a limited range and easy to be counted manually.

#### **2) Rule of thumb method**

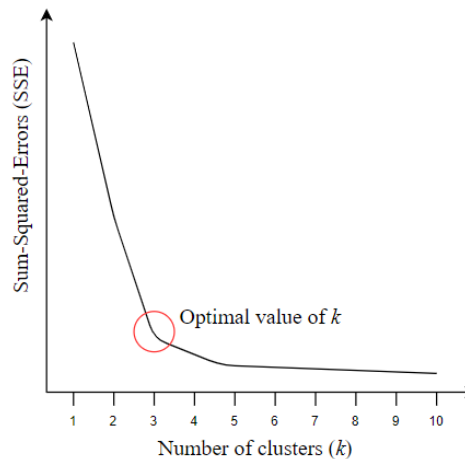
In the real world, most datasets have a great number of dimensions so the value of  $k$  is not able to predict easily. In such circumstances, the rule of thumb method is a simple way to identify the value of  $k$ . Rule of thumb method is suitable to any type of datasets and it designs the following equation to obtain the value of  $k$ :

$$k \approx \sqrt{\frac{n}{2}} \quad (2)$$

Where  $n$  is the total number of data points.

### 3) Elbow method

A traditional method to determine the value of  $k$  for  $k$ -means clustering is called as elbow method, which is based on vision of a graph based on the relationship between the sum-squared-errors (SSE) and the number of clusters. The elbow method determines the value of  $k$  based on the vision of the SSE- $k$  graph. E.g., we let the number of clusters start from  $k=2$  in order to output the SSE, then let the value of  $k$  step up by one each time and reproduce the new SSE until  $k$  reaches the given limit. From the initial stage, the value of SSE decreases strikingly and then after some point it will keep relatively stable. The value of  $k$  at this point will be closest to the actual number of the clusters. If we put SSE as Y-axis and the value of  $k$  as X-axis then we get a graph of the relationship between SSE and  $k$ , see illustration below:





**Fig. 3.** X-axis: Number of clusters; Y-axis: SSE. The line in graph looks like an elbow and one can see the point in red circle indicates the most possible actual value of  $k$ .

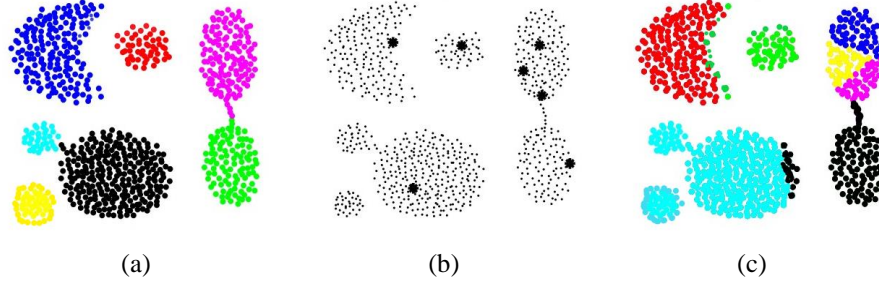
The graph above looks like an “elbow”, which is also the reason why it called elbow method. One drawback of this method is that the “elbow point” is not always explicit and sometimes there will be no or more than one “elbow point”. In such case, it is very difficult to identify the exact elbow point for proper value of  $k$ .

#### 4) Gap statistic method

The gap statistic method can be regarded as a revised version of elbow method, which is to use a statistical procedure to improve the accuracy of it. From the research in [54], Gap statistic method is more accurate and easier to find the value of  $k$  than observing the inflection point by sight of eyes in Elbow method.

##### 3.3.2 The issue of centroid initialisation

The simplest way of selecting initial centroids is to choose them at random. However, the experiment results indicate that the random initialisation of centroids puts a significant effect on the final clustering result, and sometime even causes bad or complete wrong partitions. An inappropriate initialisation of centroids often leads to converging to a local optimum and outputs incorrect clustering results [55]. Fig. 4 below shows an example of local optimum and incorrect clustering result that caused by random centroids initialisation. Therefore, how to choose the proper initial centroids is a very important problem for the  $k$ -means clustering algorithm and its variants.



**Fig. 4\***. Illustration of random centroid initialisation. (a) A given dataset with seven clusters. (b) Randomly initialised centroids. (c) Incorrect clustering result.

There are several attempts have been done and some effective algorithms have been developed to address this issue and carry out the optimal initial centroids under different circumstances, for example, the hierarchical centroid selection [56] and simple cluster seeking (SCS) [57]. We introduce them briefly as follows.

### 1) Hierarchical centroid selection

The hierarchical centroid selection [56] 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. To do this, the hierarchical centroid selection algorithm first runs basic  $k$ -means clustering multiple times with random initialisation and produces multiple groups of centroids, and then uses these groups of centroids as input data points and runs  $k$ -means clustering on it to output the final centroids. Table 3 below is the pseudo code for hierarchical centroid selection.

---

\* It is noteworthy this graph is downloaded from Internet.

**Table 3. The pseudo code of hierarchical centroid selection**


---

**Input:** data points  $\mathbf{X} = \{ \mathbf{x}_1, \dots, \mathbf{x}_n \} \in \mathbb{R}^{n \times d}$ ,  
repeat times  $p$ , clusters number  $k$ ;

---

**Output:**  $k$  centriods;

---

- 1: Randomly select  $k$  data points as centroids;
  - 2: Repeat step 1 for  $p$  times and get  $p$  groups of centroids, marked as  $\mathbf{C}_i = \{ \mathbf{c}_1, \dots, \mathbf{c}_k \mid i = [1, p] \}$ ;
  - 3: Calculate the mean value of  $\mathbf{C}_i$ ,  $\mathbf{C} = \text{mean}(\mathbf{C}_i)$ ,  $i = [1, p]$ ;
  - 4: Run  $k$ -means clustering on  $\mathbf{X}$  with  $\mathbf{C}$  as initial centroids.
- 

## 2) Simple cluster seeking (SCS)

Simple cluster seeking (SCS) [57] 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 repeats until  $k$  centroids are generated. Simple cluster seeking is also the default algorithm of centroid selection for  $k$ -means function in Matlab software suite.

### 3.3.3 The issue of similarity measurement

The third issue of  $k$ -means clustering is how 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, Pearson correlation coefficient, Jaccard similarity coefficient and averaged Kullback-Leibler divergence are also used widely.

Table 4 below shows the advantages and disadvantages of the common distance measurements above.

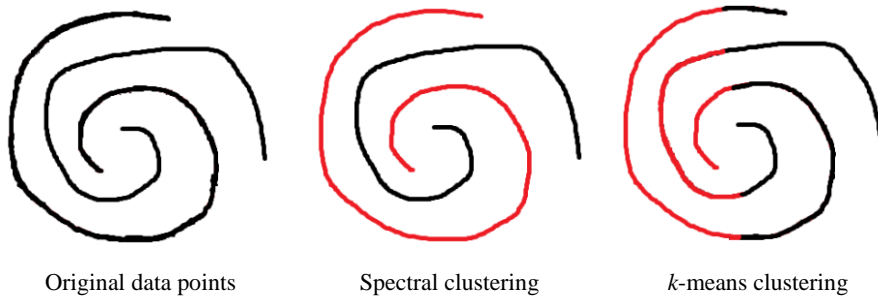
**Table 4.** Distance metrics

Distance Metric	Formula	Algorithms in which it is used	Benefits	Drawbacks	Application Area
Euclidean	$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$	- Partition based algorithms - $K$ Modes	Easy to implement and Test	- Results are greatly influenced by the largest value of variables - Poor performance for image data, document classification	- Interval data analysis in health of psychology - DNA Analysis
Manhattan	$ x_1 - x_2  +  y_1 - y_2 $	Partition based algorithms	Easily generalized to higher dimensions	Does not work well for image data, Document Classification	In Integrated Circuits
Cosine	$\cos(\theta) = \frac{A \cdot B}{\ A\  \ B\ }$	Ontology and Graph based algorithms	Handles both Continuous and categorical variables	Does not work well for nominal data	Text Mining
Jaccard Coefficient	$d_\mu(A, B) = \frac{\mu(A \Delta B)}{\mu(A \cup B)}$	Neural Network	Handles both Continuous and categorical variables	Does not work well for nominal data	Document classification

### 3.4 Spectral clustering introduction

Recently, spectral clustering [58] is becoming increasingly popular. Comparing with the traditional  $k$ -means clustering, spectral clustering is easy to implement and has excellent adaption to multiple types of datasets, as well as good efficiency and performance. Spectral clustering

always performs better than  $k$ -means clustering in various applications, especially for data points in arbitrary shapes rather than global shapes. See Fig. 5 below for detailed illustration.



**Fig. 5.** Comparison of  $k$ -means and spectral clustering results

An important difference to  $k$ -means clustering is that spectral clustering pre-processes training data points by replacing low-order relationship or original data points with high-order relationship representation [59, 60]. To implement this, spectral clustering, firstly, constructs a similarity matrix  $\mathbf{W}$ , which contains the similarity relationship between every two data points. Then spectral clustering transfers this similarity matrix into a sparse adjacent matrix by using a kernel function, which also helps to reduce the computational complexity. In the next step, spectral clustering computes the matrix Laplacian and decompose the first  $k$  eigenvectors. Finally, spectral clustering outputs the final clustering result by applying  $k$ -means clustering on the dimension-reduced matrix that consists of the first  $k$  eigenvectors. We will discuss the details in theory in the following sections.

In another words, the key idea of spectral clustering is to reformulate the problem of clustering into a problem of optimal graph cutting, i.e.,

finding a solution that makes the sum of weight of edges between different groups is minimal and the sum of weight of edges within same cluster is maximal. From most former research works, for a given set of data points  $\mathbf{X} = \{ \mathbf{x}_1, \dots, \mathbf{x}_n \} \in \mathbb{R}^{n \times d}$ , if we define  $w_{ij}$  as the similarity between any pair of data points  $\mathbf{x}_i$  and  $\mathbf{x}_j$ , a nice approach of producing similarity matrix is to construct a undirected graph  $\mathbf{G} = (\mathbf{V}, \mathbf{E})$  as the representative. In this graph, vertices (marked as  $\mathbf{v}_i$ ) are the representations of the original data points ( $\mathbf{x}_i$ ) and the edges between them denote the weight (marked as  $\mathbf{e}_i$ ). If the similarity  $w_{ij}$  between any vertices pair meets some predefined criteria then these two vertices are connected ( $w_{ij} > 0$ ).

As a result, above problem is transformed to a problem of optimal graph cutting. To solve the graph cutting issues, what we need to solve is finding portions of a graph. After this, the weight of edges between different groups is as low as possible (which means the data points across different groups are maximal dissimilar), while the weight of edges within a group is as high as possible (which means the data points within the same group are maximal similar).

### 3.4.1 Graph notation

As mentioned in the last paragraph, if we let  $\mathbf{G} = (\mathbf{V}, \mathbf{E})$  be an undirected graph, then the vertices matrix are denoted by  $\mathbf{V} = \{ \mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n \}$  and the edges matrix are denoted by  $\mathbf{E} = \{ \mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_m \}$ ,  $m = n \times (n-1)/2$ . We also define the graph  $\mathbf{G}$  is weighted and the similarity between any two data points  $\mathbf{x}_i$  and  $\mathbf{x}_j$  as a non-negative scalar so  $w_{ij} \geq 0$ . The similarity matrix  $\mathbf{W}$  of graph  $\mathbf{G}$  can be obtained as below:

$$\mathbf{W} = (w_{ij}) \quad (i, j \in [1, n]) \quad (3)$$

Where  $\mathbf{w}_{i,j} = \mathbf{w}_{j,i}$  because  $\mathbf{G}$  is undirected, and  $\mathbf{w}_{i,j} = 0$  if two vertices, e.g.,  $\mathbf{v}_i$  and  $\mathbf{v}_j$ , are not connected. When the similarity measurement is Euclidean distance, the similarity matrix is defined as:

$$w_{i,j} = \sqrt{\sum_{t=1}^d (x_{i,t} - x_{j,t})^2} \quad (i, j \in [1, n], t \in [1, d]) \quad (4)$$

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

Form the equation (4) above we can see the dimensions number of similarity matrix  $\mathbf{W}$  is  $n$  by  $n$ , therefore its size will be very huge and the computational complexity will be extreme high when the dataset is massive. To address this problem and improve the efficiency, after the similarity matrix has been produced spectral clustering transfers this similarity matrix into a sparse matrix by using designated algorithms and kernel functions. In this thesis, we term this sparse similarity matrix as adjacent matrix and mark as  $\mathbf{A}$ .

In spectral clustering more than half calculations apply on the adjacent matrix  $\mathbf{A}$ , hence the quality of the adjacent matrix will pose a significant influence on both the final clustering result and the processing time. It is very important and essential to take into account as many factors of the dataset as possible when constructing the adjacent matrix. Besides, one should evaluate the adjacent matrix carefully and accurately on a case-by-case basis. In the past decades, many efforts have been done on constructing the similarity graph and adjacent matrix, as a sequence, there are various algorithms have been developed. Among these algorithms

three are common and widely used, i.e.,  $\epsilon$ -neighbourhood graph,  $k$ -nearest neighbour graph, and full-connected graph [60]. We introduce them in details as follows.

Firstly, the  $\epsilon$ -neighbourhood graph algorithm connects two neighbored vertices (marked as  $e_{i,j}=1$ ) if the pairwise distance is less than  $\epsilon$ , and does not connect otherwise (marked as  $e_{i,j}=0$ ), where  $\epsilon$  is a given threshold depending on the properties of the datasets. This makes all edges of a graph roughly have the same value (i.e., the value of  $\epsilon$ ) and leads to an unweighted graph, because there is no more information of the dataset incorporated to the graph during the construction.

Secondly, the  $k$ -nearest neighbour graph algorithm connects  $\mathbf{v}_i$  and  $\mathbf{v}_j$  if  $\mathbf{v}_j$  is in the range 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. So far, there are two common ways to make the graph as undirected. The first one is to disregard the directions of the edges simply, which is usually termed as the  $k$ -nearest neighbour graph. The second way is to only connect vertices  $\mathbf{v}_i$  and  $\mathbf{v}_j$  if both  $\mathbf{v}_i$  is one of the  $k$ -nearest neighbours of  $\mathbf{v}_j$  and  $\mathbf{v}_j$  is also among the  $k$ -nearest neighbours of  $\mathbf{v}_i$ , which is usually named as mutual  $k$ -nearest neighbour graph. The edges need to be weighted in both cases by the similarity vectors of the adjacent data points after connecting the appropriate vertices.

Thirdly, the full-connected graph algorithm simply connects all the vertices between each other with its similarity scalar, which is computed by similarity function. In this case, it is required that the similarity function itself is able to encode the major local neighbourhood relationships,



because the graph is expected to be able to model the local neighbourhood relationships. As we mentioned in chapter 2.6, popular kernel functions include fisher kernel, graph kernel, kernel smoother, polynomial kernel, Gaussian kernel [61], sigmoid kernel, radial basis function kernel (RBF) and string kernel, among these the Gaussian kernel function already encodes the mainly local neighbourhood relationships so it has been using widely. Due to the demonstration above, in this paper, we choose the full-connected graph to construct the adjacent matrix to represent the original data points. Additionally, we choose the Gaussian kernel as the similarity function.

Moreover, another important matrix involved during the spectral clustering is the degree matrix, which is usually marked as  $\mathbf{D}$ . The degree matrix is defined as a diagonal matrix whose elements are the degree of each vertex. As we know, the degree of a vertex can be calculated as:

$$d_i = \sum_{j=1}^n a_{i,j} \quad (i, j \in [1, n]) \quad (5)$$

Where  $a_{i,j}$  denotes the  $i$ -th row and  $j$ -th column element of the adjacent matrix  $\mathbf{A}$ . Therefore, the degree matrix  $\mathbf{D}$  is defined as the following:

$$\mathbf{D} = (\mathbf{d}_{i,i}) \quad (i \in [1, n]) \quad (6)$$

In our second proposed clustering method, we compute the weight of features based on the degree matrix above.

### 3.4.2 Matrix Laplacian

The second step of spectral clustering is to produce the Laplacian matrix, marked as  $\mathbf{L}$ . Laplacian matrices are deprived from the spectral graph

theory and have multiple variants. In this thesis, we mainly introduce two versions of Laplacian matrix: Unnormalised Laplacian matrix and Normalised Laplacian matrix [60].

Then unnormalised Laplacian matrix is defined as:

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \quad (7)$$

While the normalised Laplacian matrix is defined as:

$$\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-\frac{1}{2}} \quad (8)$$

Where  $\mathbf{D}$  is the degree matrix, which is defined in Eq. (6).

Finally, spectral clustering conducts dimension reduction by selecting  $k$  eigenvectors of  $\mathbf{L}$  to construct matrix  $\mathbf{U}$ , and then conducts  $k$ -means clustering on matrix  $\mathbf{U}$  to output the final clustering result. We list the details of spectral clustering in Table 5.

**Table 5.** The pseudo code of spectral clustering

---

<b>Input:</b> data points $\mathbf{X} = \{ \mathbf{x}_1, \dots, \mathbf{x}_n \} \in \mathbb{R}^{n \times d}$ ; the cluster number $k$ .
<b>Output:</b> the cluster indicators of all data points and centroids $\mathbf{C}$ .
<b>1:</b> Compute the similarity matrix $\mathbf{W}$ of $\mathbf{X}$ by Eq. (3);
<b>2:</b> Compute the matrix Laplacian $\mathbf{L}$ by Eq. (8);
<b>3:</b> Compute the first smaller $k$ eigenvectors of $\mathbf{L}$ , marked as $\mathbf{E} = \{ \mathbf{e}_1, \dots, \mathbf{e}_k \}$ ;
<b>4:</b> Construct matrix $\mathbf{U}$ , where $\mathbf{U} = \mathbf{E}^T$ , $\mathbf{U} \in \mathbb{R}^{n \times k}$ ;
<b>5:</b> Run $k$ -means clustering on $\mathbf{U}$ to output the cluster result $\mathbf{C}$ .

---

### **3.5 Summary of $k$ -means clustering and SPCL**

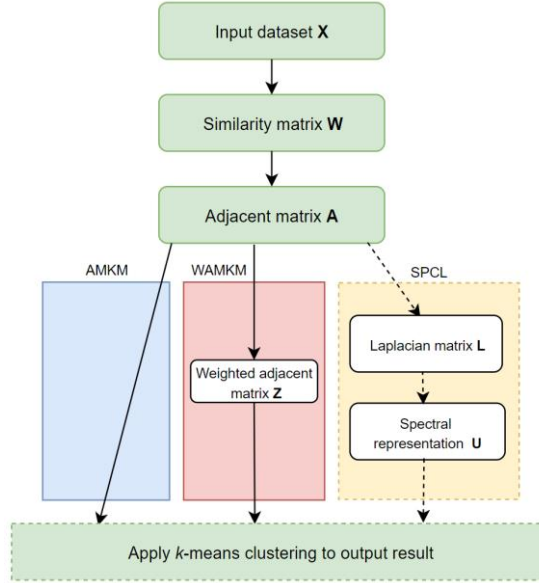
In this chapter, we introduced the  $k$ -means clustering and spectral clustering algorithms in details, including the objection functions of  $k$ -means clustering, implementations, advantages and disadvantages. We also discussed the common solutions for their shortcomings and the possible improvements. In the following chapter, we will introduce our proposed clustering methods based on these discussions.

## Chapter. 4. Proposed Methods

In the previous chapters, we introduced the fundamental concepts and algorithms involved in machine learning areas. We then introduced the common and widely used clustering algorithms such as partition based clustering, density based clustering, hierarchy based clustering, graph based clustering, kernel based clustering and grid based clustering, with their implementations and typical representing algorithms briefly. Furthermore, we also discussed the famous and popular clustering algorithms, e.g.  $k$ -means clustering and spectral clustering in details, followed by their advantages and disadvantages.

Although spectral clustering algorithm has a number of benefits than  $k$ -means clustering algorithm, it still incorporates  $k$ -means clustering as the final step to output the result. So spectral clustering is not able to avoid suffering from the limitations of  $k$ -means clustering. Furthermore, spectral clustering requires a similarity matrix whose dimension is  $n$  by  $n$ . For massive datasets, this similarity matrix is very large and always results in extreme high time cost when decomposing the eigenvectors and eigenvalues of the matrix Laplacian. Therefore, the computational complexity is huge and unacceptable for large datasets. Hence, in this thesis, we focus on both improving the clustering accuracy and reducing the computational complexity by proposing two novel clustering methods. These two methods focus on optimising the clustering mechanism and constructing efficient similarity matrices. 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:



**Fig. 6.** 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.

#### 4.1 AMKM

In this section, we introduce our first proposed clustering method, i.e., adjacent matrix based  $k$ -means clustering method (AMKM). The initial step in conducting spectral clustering is to construct the similarity matrix by transferring the data points into an undirected graph  $G = (\mathbf{V}, \mathbf{E})$ , where  $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$  denotes the vertices, and  $\mathbf{E} = \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_m\}$  ( $m = n \times (n-1)/2$ ) denotes the edges between vertices. The undirected graph is abstracted and represented by the similarity matrix  $\mathbf{W} = (w_{i,j})_{i,j=1}^n$ , where

$w_{i,j} \geq 0$  means the similarity between  $\mathbf{x}_i$  and  $\mathbf{x}_j$  under a given distance metric. The adjacent matrix  $\mathbf{A}$  is constructed based on  $\mathbf{W}$  by the following methods.

When constructing the adjacent matrix the most important problem is to model the relationships of local neighbourhood between data points. In the past decades, researchers have paid much effort on constructing the adjacent matrix, including  $\epsilon$ -neighbourhood graph,  $k$ -nearest neighbour graph, and fully connected graph [60]. For example, the  $\epsilon$ -neighbourhood graph connects two neighboured vertices (i.e.,  $e_{i,j}=1$ ) if the pairwise distance is less than a given threshold  $\epsilon$ , otherwise, it does not connect them (i.e.,  $e_{i,j}=0$ ). This makes all edges of a graph roughly have the same value (i.e.,  $\epsilon$ ) 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 [61] and Sigmoid kernel. When a Gaussian kernel function is used, the adjacent matrix is defined as follows:

$$a_{i,j} = e^{-\left(\frac{\|\mathbf{w}_i - \mathbf{w}_j\|_2^2}{2\sigma^2}\right)} \quad (i, j \in [1, n]) \quad (9)$$

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 6.

**Table 6.** The pseudo code of our proposed AMKM method

---

<b>Input:</b> data points $\mathbf{X}=\{ \mathbf{x}_1, \dots, \mathbf{x}_n \} \in \mathbb{R}^{n \times d}$ ; the cluster number $k$ .
<b>Output:</b> the cluster indicators of all data points and centroids $\mathbf{C}$ .
<b>1:</b> Calculate the similarity matrix $\mathbf{W}$ of $\mathbf{X}$ by Eq. (3);
<b>2:</b> Calculate adjacent matrix $\mathbf{A}$ by Eq. (9);
<b>3:</b> Run $k$ -means clustering on $\mathbf{A}$ to output $\mathbf{C}$ .

---

The experiment results show that our AMKM clustering algorithm outperforms the comparison clustering algorithms on more than 90% of the selected datasets. We illustrate the experiment results of AMKM and the comparisons algorithms in chapter 5, followed by analysing the performance differences between different datasets. We implemented our clustering algorithm above in Matlab 2019, and we recorded the code in Appendix A.

## 4.2 Weighted AMKM

In the last section, we introduced our improved  $k$ -means clustering method, i.e. adjacent matrix based  $k$ -means clustering method. In generally, when conducting clustering most algorithms consider the features of datasets on an equal-weight basis, which means that each feature is as same as important. 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. Hence, this assumption of equal-weight normally affects the clustering results seriously since features are not likely to have equal importance in the real world applications. For example, we consider a database of adult dog species, which consists of four features, i.e., body size, body colour, tail shape and tail length. From the perspective of zoology, it is obvious that the feature of body size is more important than the feature of body colour. Since the body colour is a very common attribute for dogs, and it is very likely that dogs are able to have the same colour even they are from different species. Therefore, a white dog and a black dog (the similarity of body colour is very low) with the same body size (the similarity of body size is very high) are likely belong to the same species, but the inverse is not. This means the importance of body size is higher than the importance of body colour, i.e., features have different importance or weight [62].



From this perspective, we should give more priority on the feature that has more weight when constructing the adjacent matrix. In this chapter, we introduce our second clustering method – weighted adjacent matrix based  $k$ -means clustering method (WAMKM), which takes the weight of features into account, followed by its implementation in details. The weight is a term in statistics. In the data science area, weight is considered as a factor that is to measure how important a feature or attribute is when comparing with others in the same dataset.

In our WAMKM, we consider the importance of different features and compute the weight of each feature. When doing features extraction and constructing the similarity matrix, a common way used is that for each data point representation in the adjacent matrix  $\mathbf{A}$ , each feature is represented by a numeric scalar. It has been discussed in the above paragraph that different feature always put different influence on the clustering result. Therefore, we calculate the weight of each data point representation in the adjacent matrix  $\mathbf{A}$  and construct a weighted adjacent matrix (in this thesis we denote it as  $\mathbf{Z}$ ). Since there is no prior information of weights of the data points is given, so in our method 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  $\mathbf{d}$  ( $\mathbf{d} = \{d_1, \dots, d_n\}$ ), where  $d_j$  is the summation of all elements in the  $j$ -th column of  $\mathbf{A}$ , and then we normalise the weight vector by:

$$\mathbf{h} = \frac{d_j}{\sum_{j=1}^n d_j} \quad (j \in [1, n]) \quad (10)$$

Eq. (6) makes the sum of all elements in  $\mathbf{h}$  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  $\mathbf{Z}$  by applying the weight vector  $\mathbf{h}$  on each data point in adjacent matrix  $\mathbf{A}$ :

$$z_{i,j} = a_{i,j} \times h_j \quad (i, j \in [1, n]) \quad (11)$$

Finally, after the weighted adjacent matrix  $\mathbf{Z}$  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 7.

**Table 7.** The pseudo code of our proposed WAMKM method

<b>Input:</b> data points $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^{n \times d}$ ; the cluster number $k$ .
<b>Output:</b> the cluster indicators of all data points and centroids $\mathbf{C}$ .
<b>1:</b> Produce the similarity matrix $\mathbf{W}$ of $\mathbf{X}$ by Eq. (3);
<b>2:</b> Calculate the adjacent matrix $\mathbf{A}$ by Eq. (9);
<b>3:</b> Calculate the weight vector $\mathbf{h}$ by Eq. (10);
<b>4:</b> Calculate the weighted adjacent matrix $\mathbf{Z}$ by Eq. (11);
<b>5:</b> Run $k$ -means clustering on $\mathbf{Z}$ to output $\mathbf{C}$ .

The experiment results indicate that our WAMKM clustering algorithm performs better than AMKM clustering algorithm on most selected datasets, in terms of all the three evaluation metrics. We illustrate the experiment results of WAMKM and the comparisons algorithms in chapter 5, followed by our analysis in details. We also implemented the WAMKM in Matlab 2019, and we recorded the code in appendix B.

### 4.3 Summary of AMKM and WAMKM

In this chapter, we introduced our two novel clustering methods, i.e., adjacent based  $k$ -means clustering method and weighted adjacent based  $k$ -means clustering method. The first method runs  $k$ -means clustering on the adjacent matrix directly, while the second method is a revised version of the first one, and takes into account the weight of features. Both methods outperformed the comparison clustering algorithms in terms of all the three evaluation metrics in our experiment on the selected datasets.

## Chapter. 5. Experimental Analysis

In this thesis, we selected twelve datasets from various sources to evaluate our two methods, comparing with the spectral clustering,  $k$ -means clustering and  $k$ -means++ clustering, in terms of three evaluation metrics which are ACC, NMI and Purity. We describe these datasets and evaluation metrics in details as bellows.

### 5.1 Datasets

The selected 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 introduce these twelve datasets with their details as below:

- 20news consists of 3970 samples distributed in four classes and each sample consists of 8014 features.
- Binalpha consists of 1404 samples distributed in 36 classes and each sample consists of 320 features.
- Australian Credit Approval consists of 690 credit card applications within two classes, each sample consists of 14 attributes.
- Website Phishing consists of 1353 samples with nine features for each. There are three classes which are 548 legitimate websites, 702 phishing URLs and 103 suspicious URLs.
- Dexter consists of 300 samples with 20000 features for each. This dataset has two classes.

- Diabetes is a collection of “Diabetes 130-US hospitals for years 1999-2008 Data Set” which consists of medical data of clinical care at 130 hospitals in US integrated delivery across 10 years. This dataset has eight features.
- Coil20Data consists of 1440 samples distributed in 20 classes and each sample has 1024 features.
- Cardiotocography is a collection of 2126 fetal cardiotocograms samples and 41 diagnostic features for each sample. This dataset can be used either for 10-class or 3-class experiments. In our experiment, we select three classes.
- Spambase consists of two categories, i.e. spam e-mails and non-spam e-mails. The spams were from postmasters and individuals marked as spam, while non-spam e-mails were from normal personal or work addresses. There are 57 features for each example in this dataset.
- Parkinson speech consists of 1040 samples of voice recording and each sample consists of 28 features. Moreover, the features include multiple types of sound recordings from male and female persons.
- Solar flare consists of 1066 samples distributed in six classes and each sample consists of 12 features.
- German credit data consists of 1000 samples distributed in two classes and each sample consists of 23 features.

We summarise the datasets used with their details in Table 8.

**Table 8.** Summary of the datasets used in this paper

<b>Datasets</b>	<b>Samples</b>	<b>Features</b>	<b>Classes</b>
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

## 5.2 Comparison algorithm

In this thesis, we use the clustering algorithms below as comparison algorithms.

- $k$ -means clustering is the most widely and commonly used clustering algorithm, which aims to group the data points as  $k$  clusters where the data points that belong to a 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 revised version 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) [60] is a widely used variant of the spectral clustering algorithms. Specifically, it applies  $k$ -means clustering on the normalised eigenvector matrix by normalizing the row sum to have the norm of 1.

### 5.3 Evaluation Method

In our experiment, we use the 10-fold cross validation method [63] to evaluate all the algorithms. Specifically, for a given dataset, we first divide it into ten subsets randomly, and then we use nine subsets as input to run our clustering methods and use the remaining one subset as ground truth to test the clustering results. When comparing the clustering results of our methods with the ground truth, an important thing is the data space. In spectral clustering, AMKM and WAMKM, the original data points have been transformed into a new data space in which the high-order similarity replaces the low-order similarity. Therefore, when calculating the evaluation metrics the ground truth also required to be trans-

formed into high-order similarity space, respectively. For  $k$ -means clustering this is not required since the clustering process is done on the original dataset itself.

#### 5.4 Parameter Setting

When constructing the adjacent matrix, the adjustable parameter  $\sigma$  ( $\sigma$ ) plays a vital influence on the performance of kernel function and the clustering results, hence it should be tuned carefully at hand [64]. There is no universal method in theory explains how to choose  $\sigma$  for all datasets. However, from former experience we know that the parameter  $\sigma$  governs the connectedness between data points and is different for every dataset, so it is closely related to the dataset itself. In our experiment, we tested a range of values of parameter  $\sigma$  then selected a model that applies for all datasets. Specifically, we tested the parameter  $\sigma$  in the range of  $\sigma \in [10^{-5}, \dots 10^{14}]$  on all datasets, and finally we selected the mean value of the similarity matrix  $\mathbf{W}$  as  $\sigma$  for evaluation:

$$\sigma = \text{mean}(\mathbf{W}) \quad (12)$$

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

#### 5.5 Evaluation Measurement

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) [65]. We report the definitions of the involved evaluation metrics as below.

Accuracy (ACC) is defined as:



$$ACC = \frac{N_{cor}}{N} \quad (13)$$

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 [66]. It is defined as:

$$NMI = 2 \frac{M(X_i, X_j)}{E(X_i) + E(X_j)} \quad (14)$$

Where  $M(X_i, X_j)$  is the 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 with the ground truth. It is defined as:

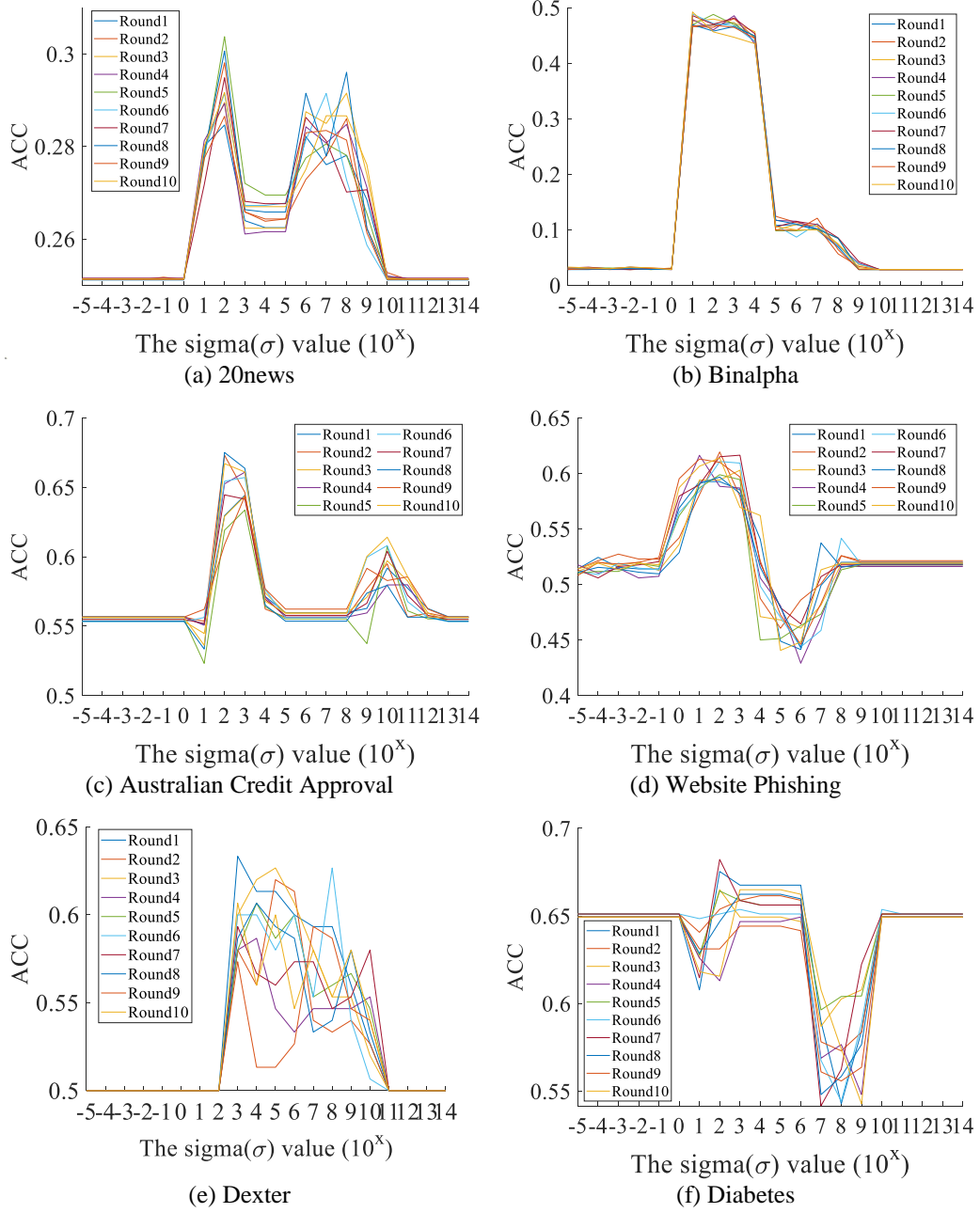
$$PUR = \sum_{i=1}^k \frac{S_i}{n} P_i \quad (15)$$

Where  $k$  is number of clusters and  $S_i$  is the number of data points of the  $i$ -th cluster.  $P_i$  denotes the distribution of correctly partitioned data points in all clusters [65].

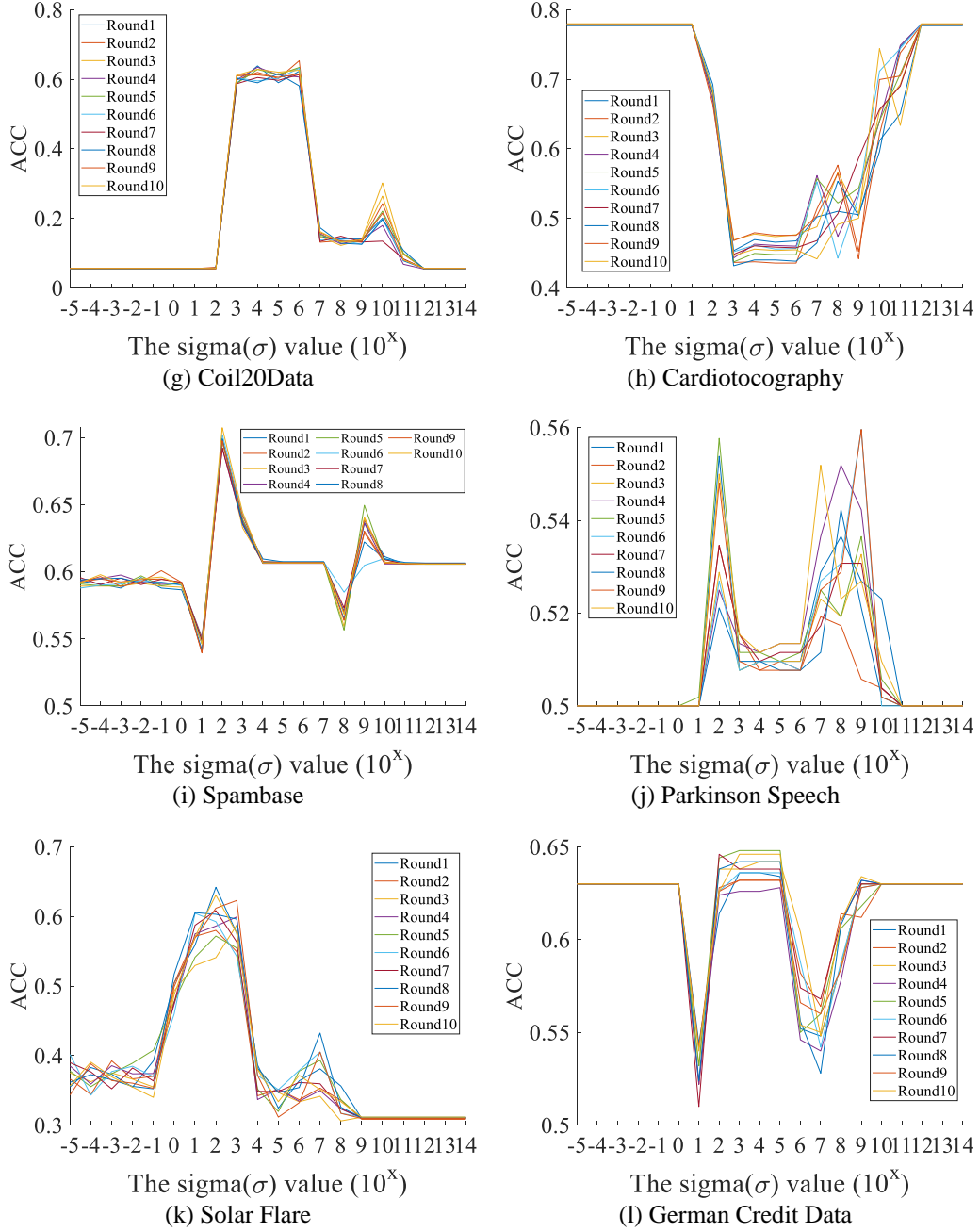
## 5.6 Experiment Result

Fig. 7a and Fig. 7b show the experiment results of ACC of our proposed WAMKM method under different settings of parameter  $\sigma$  on each dataset, and Fig. 8a and Fig. 8b are the results for AMKM.

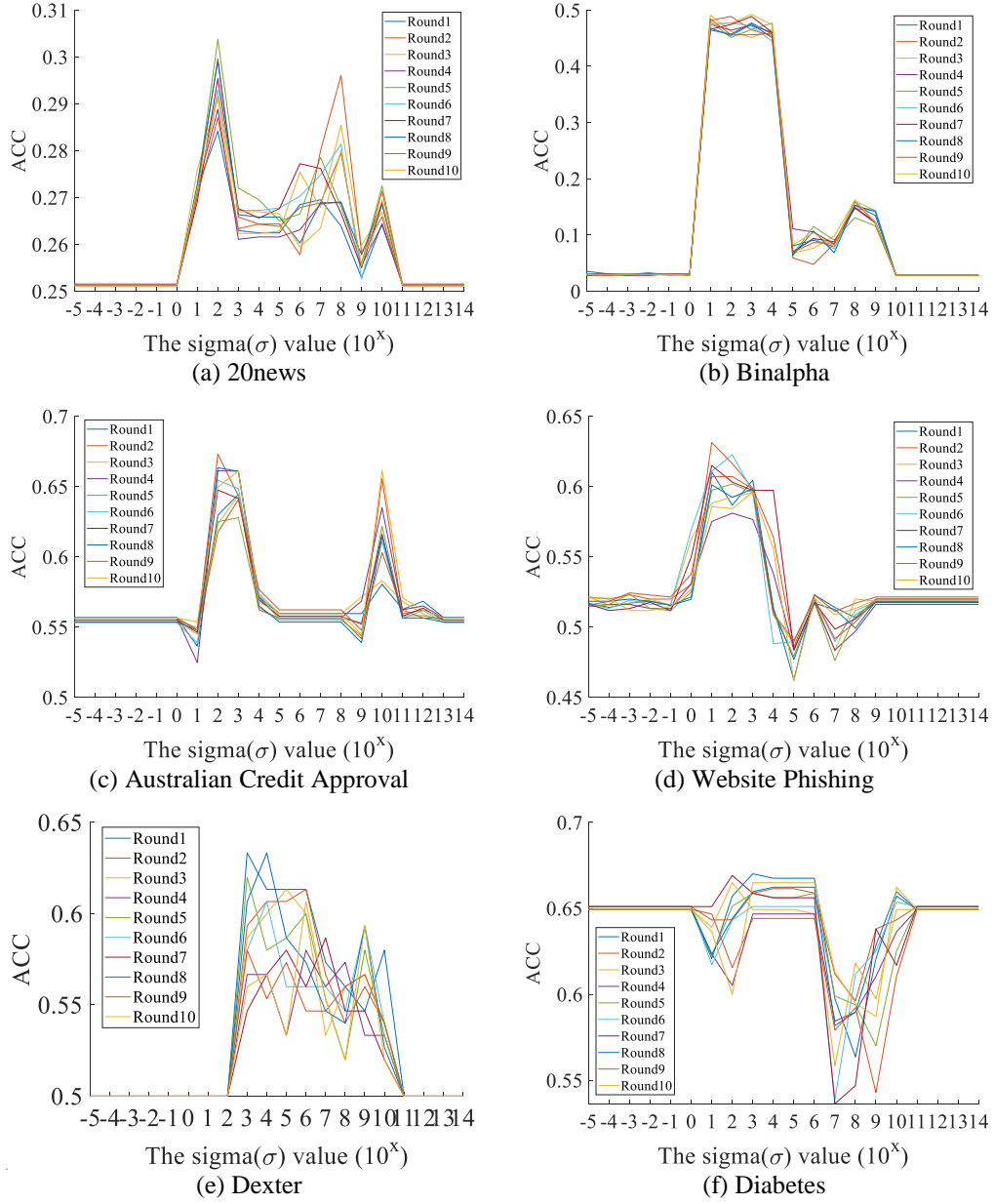
Figs. 9-11 show the results of ACC, NMI and PUR in each iteration on all 12 datasets, and Fig. 12 summarises the results of Figs. 9-11.



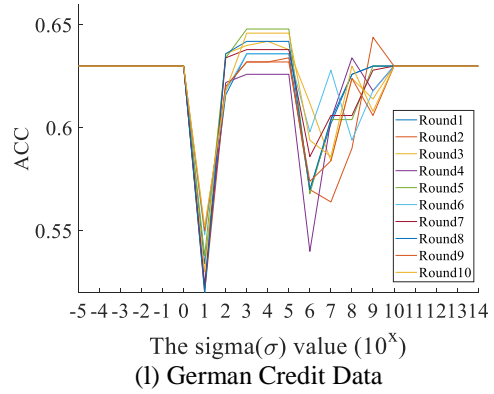
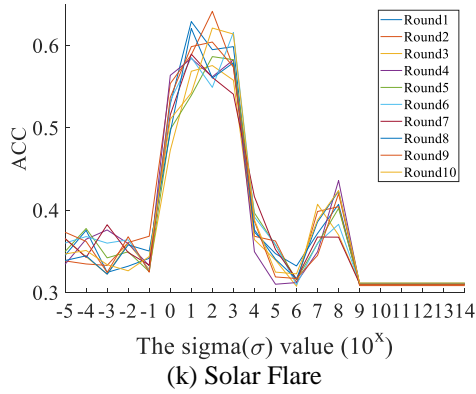
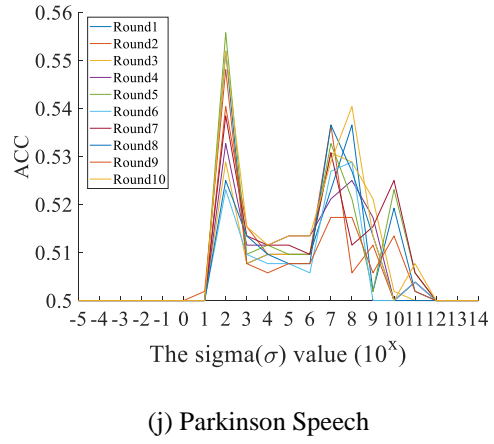
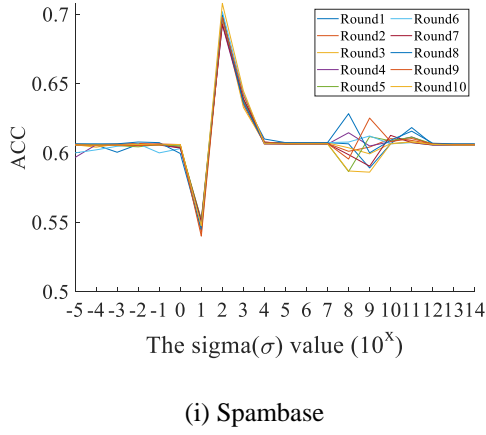
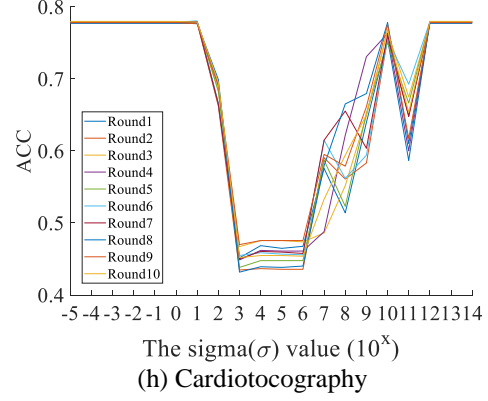
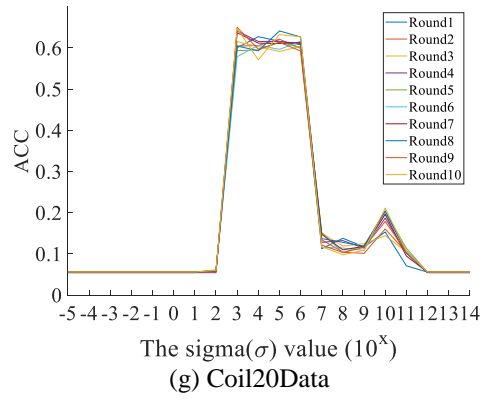
**Fig. 7a. ACC trends of our WAMKM emthod with different  $\sigma$  values.**



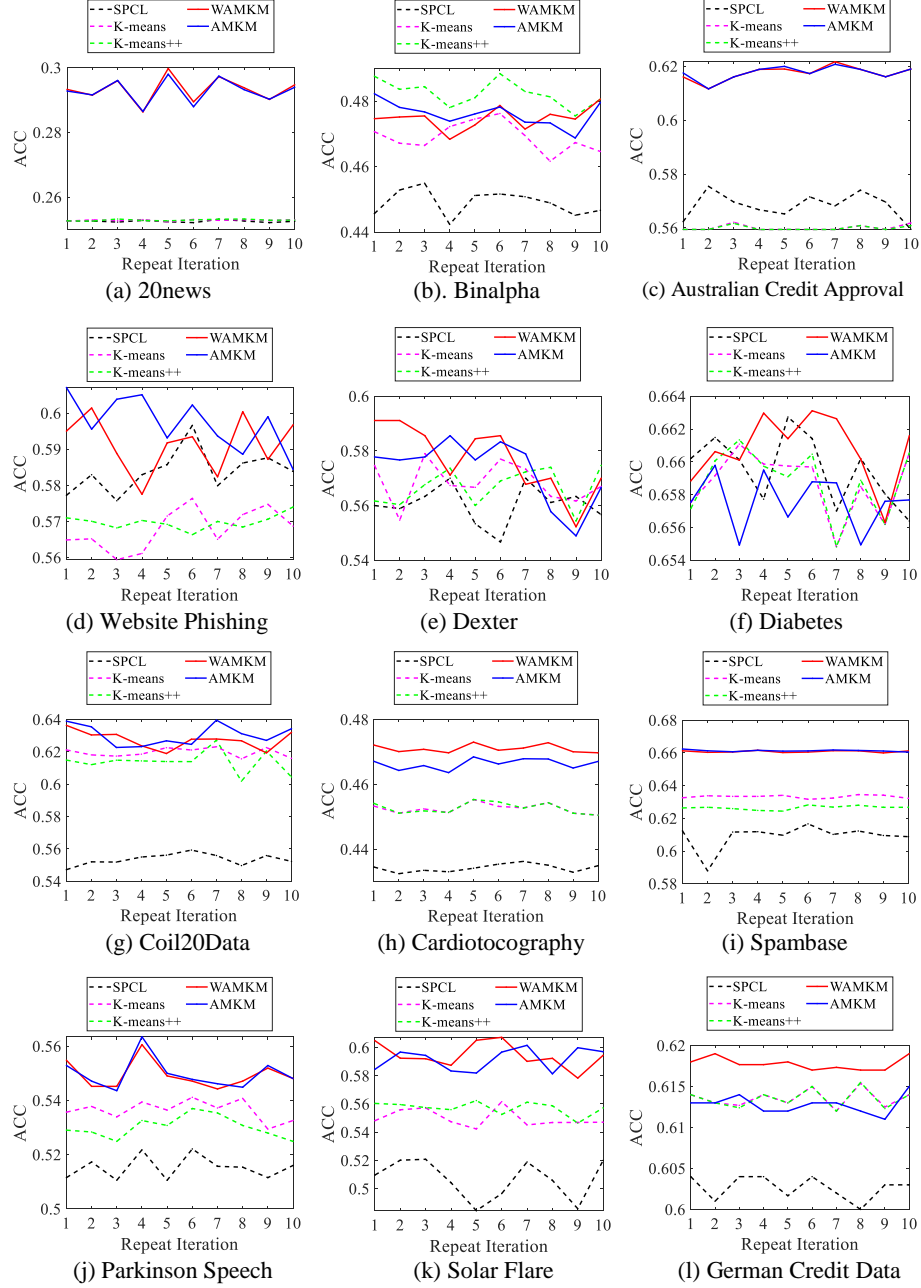
**Fig. 7b. ACC trends of our WAMKM emthod with different  $\sigma$  values.**



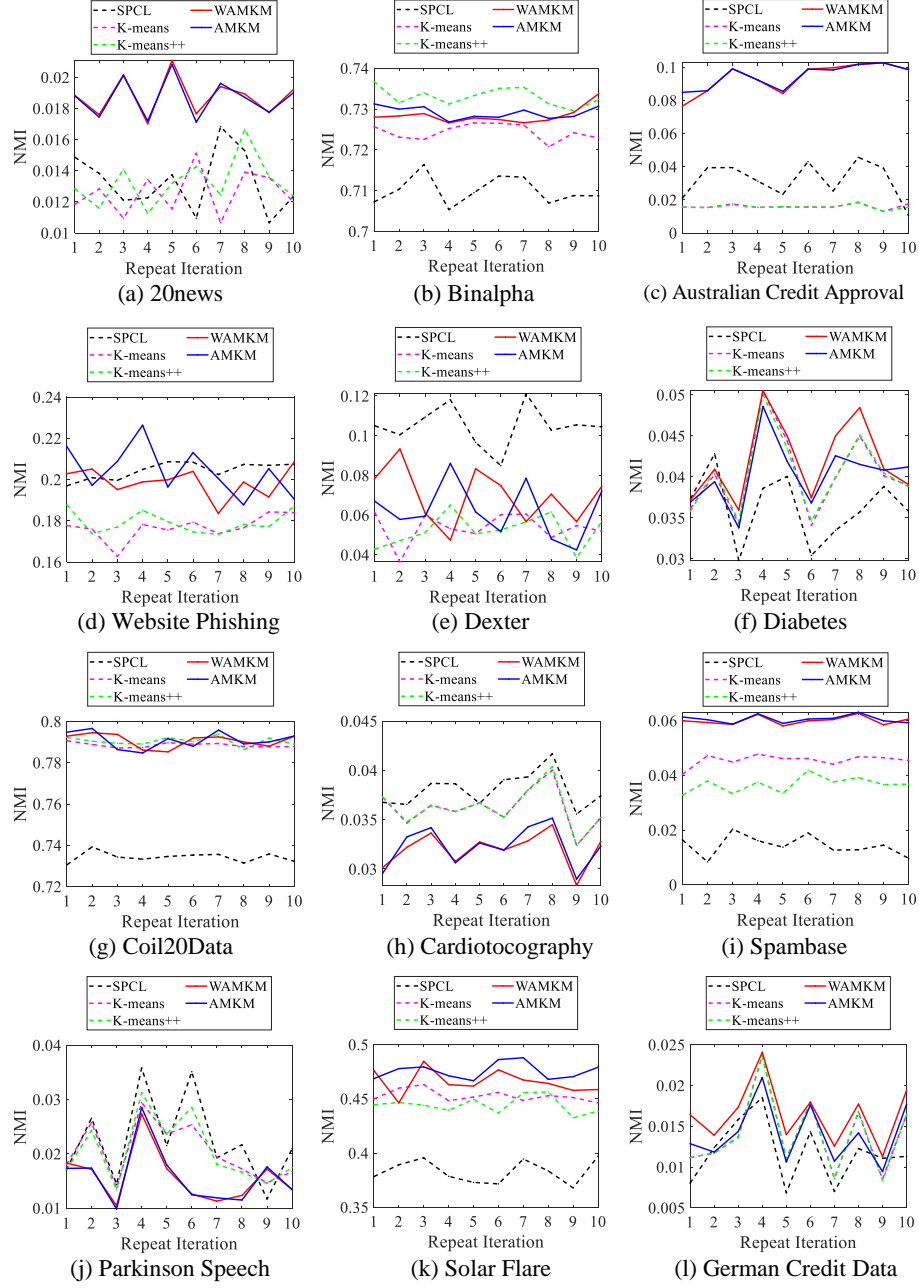
**Fig. 8a. ACC trends of AMKM with different  $\sigma$  values.**



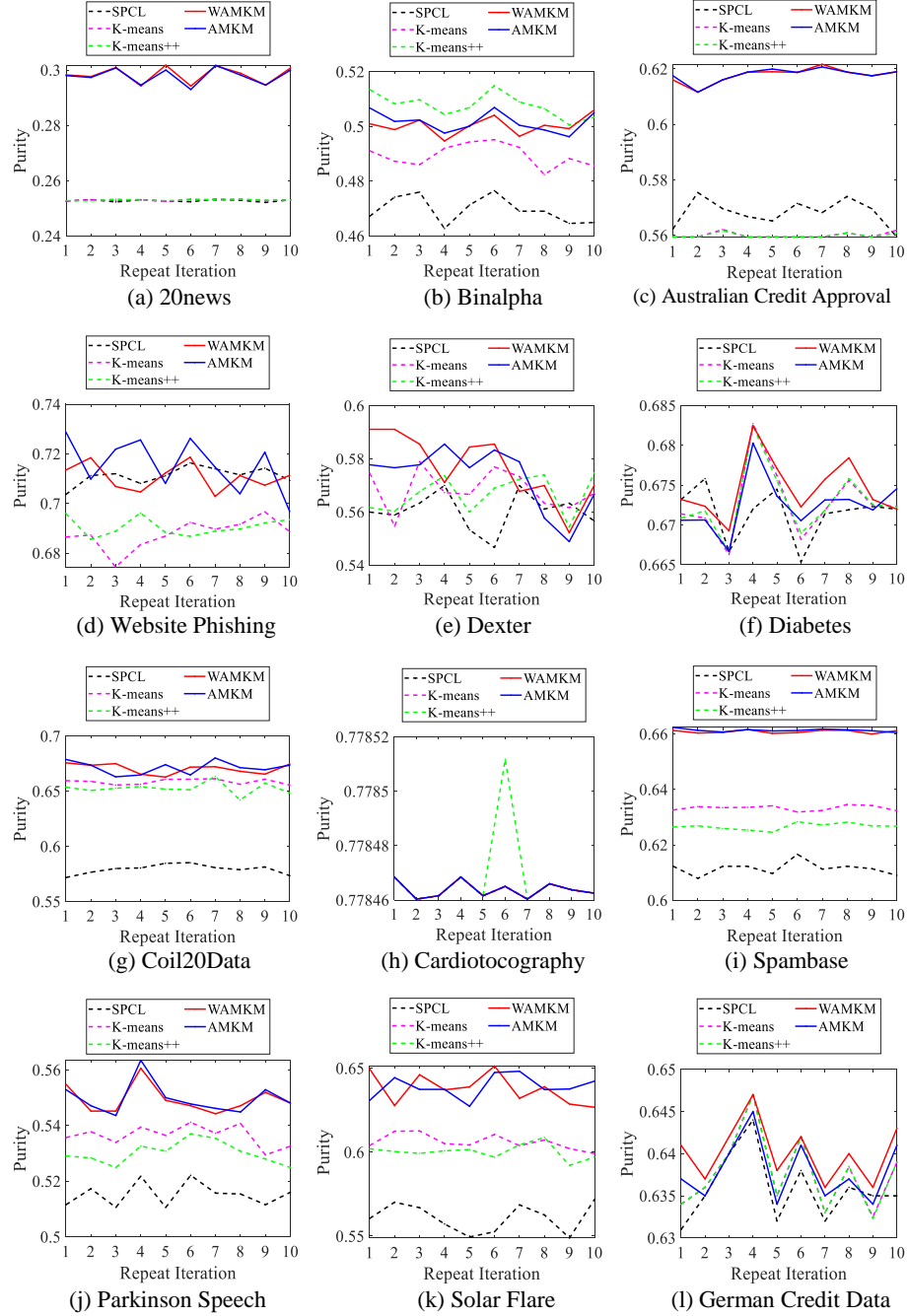
**Fig. 8b. ACC trends of AMKM with different  $\sigma$  values.**



**Fig. 9. ACC variations of all methods in each iteration of every dataset.**



**Fig. 10. NMI variations of all methods in each iteration of every dataset.**



**Fig. 11. PUR variations of all methods in each iteration of every dataset.**



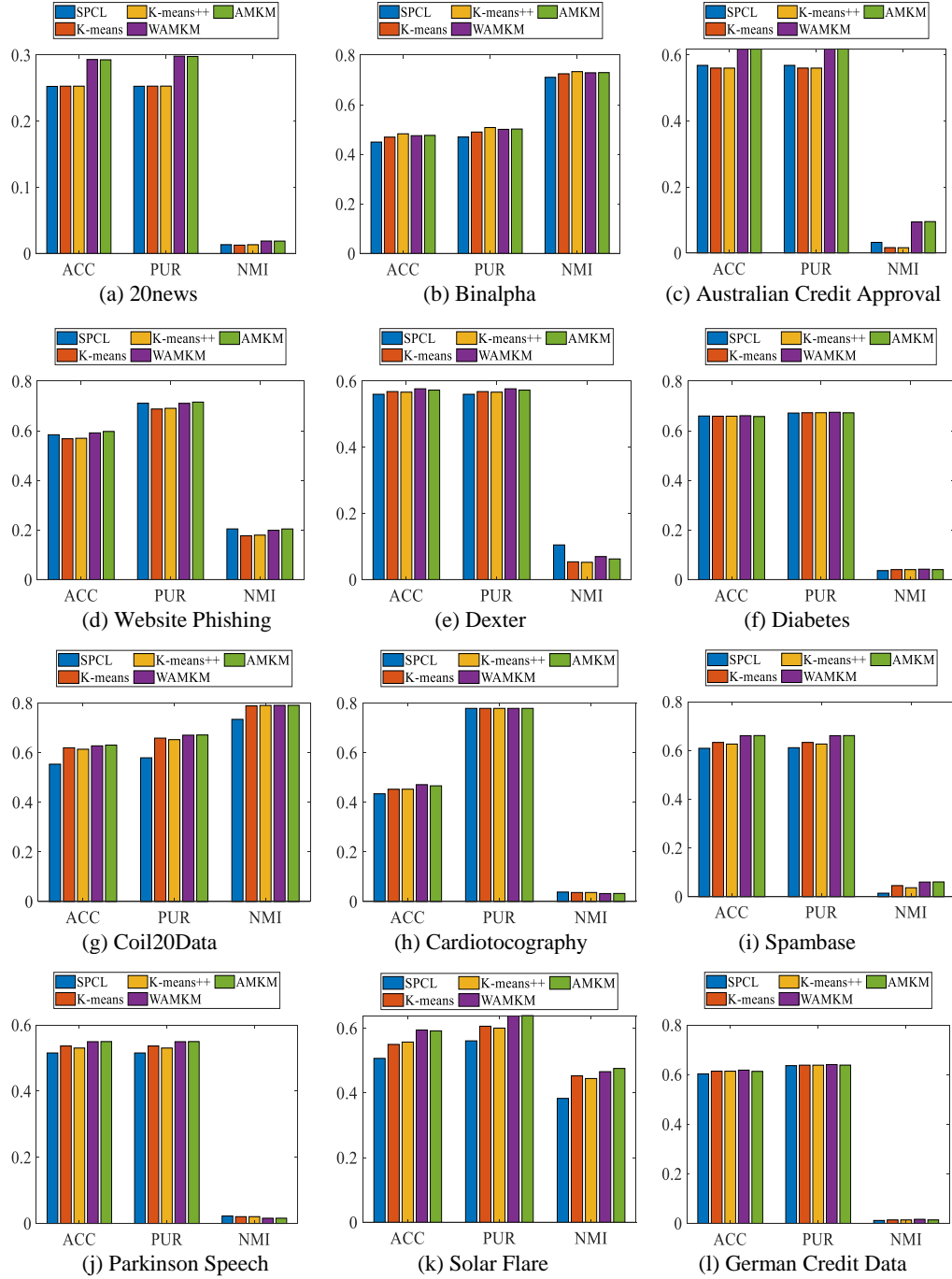


Fig. 12. The summarised results of all methods on every dataset.

### 5.7 Result analysis

Based on our experimental results in Figs. 7-11, we have the following observations.

First, our proposed methods are sensitive to the setting parameter  $\sigma$ , which controls the similarity between two data points. For example, the ACC results first keep stable while varying the value of  $\sigma$  from  $10^{-5}$  to  $10^0$ , and then begin increasing gradually until arriving their peaks, i.e.,  $10^{10}$  for the value of  $\sigma$  on some datasets, such as 20news, Binalpha, Australian Credit Approval, Coil20Data, Parkinson Speech and Solar Flare. The ACC results of show the fluctuation trends when  $\sigma$  is between  $10^0$  and  $10^{10}$ , and then keep stable while the value of  $\sigma$  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. 8a and Fig. 8b. Moreover, our proposed WAMKM method is more sensitive to the value of the parameter  $\sigma$ , compared with our proposed AMKM method. In nutshell, the ACC results on our selected datasets vary while the value of parameter  $\sigma$  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  $\sigma$  is too small or too large. Hence, it is essential to tune the value of parameter  $\sigma$  carefully and accurately. Moreover, to archive the best clustering performance, different datasets should use different ranges of  $\sigma$ .

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 with spectral clustering,  $k$ -means and  $k$ -means++ 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 with the use of spectral representation of SPCL and the use of original features in both  $k$ -means and  $k$ -means ++ clustering. It implies that representation learning is very important for clustering analysis, which was demonstrated in the literature [22] [40].

Last but not least, our proposed WAMKM method has no significant improvements, compared with 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  $\sigma$ . However, our proposed WAMKM method is more sensitive than our proposed method AMKM, in terms of the variations of the parameter  $\sigma$ .

## Chapter. 6. Conclusion and Future Work

### 6.1 Conclusion

In this thesis, we have proposed two clustering methods to address the issues of previous  $k$ -means clustering. To achieve this, we first devised an adjacent matrix and a weighted adjacent matrix, respectively, followed by conducting  $k$ -means clustering on the resulted adjacent matrix. 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. As a result, our proposed clustering methods outperform the comparison algorithms in our experiments.

### 6.2 Future Work

However, we found that the experiment results of our proposed methods are sensitive to parameter  $\sigma$ , which is used to construct the adjacent matrix. Inappropriate selected parameters always result in bad or complete wrong clustering results. This means that our proposed clustering methods are data-driven and their performance varies on different types of datasets. Hence, in our future work, we will extend our research to dynamically select suitable parameters and develop a novel algorithm to select the best parameter  $\sigma$  based on the dataset itself, instead of the mean value of similarity matrix used in this thesis.

## Appendices

We implemented our clustering methods in Matlab 2019a, and we attach our implementing code in this part.

### Appendix A. Matlab code for AMKM

```
function [ACC, NMI, Purity, ARI, telabel, center] = mySPCL2(Xtr, Xte, gnd, opt)
%% example clear; clc; opt.sigma = 1:5; opt.k = 10; opt.i = 1; opt.splType = 2;
%% [ACC, NMI, Purity, ARI, telabel, center] =
%% mySPCL2(rand(3000,50), rand(500,50), [ones(300,1); 2*ones(200,1)], opt)

%% calculate the similarity matrix

W = EuDist2(Xtr, Xtr);

switch opt.para_tuning
% parameter tuning
case 1
    sigma = opt.sigma(opt.para_pos);
% no parameter tuning
case 0
    sigma = mean(mean(W));
end

% calculate the adjacent matrix
W = exp(-W.^2 ./ (2*sigma^2));
% process the test data to match with the training data space
Xte = EuDist2(Xte, Xtr);
Xte = exp(-Xte.^2 ./ (2*sigma^2));

for j=1:opt.kmeans_repeat
    %% run k-means clustering on adjacent matrix directly
    [~, center] = kmeans(W, opt.k, 'start', 'cluster');
    % process the test data to match with the training data space
    % dist_all denotes distances of all data points in test data to clusters centres;
    dist_all = EuDist2(center, Xte);
    % select the minimum distance to a centre and assign class label to this centre;
    [~, telabel] = min(dist_all);
    telabel = telabel';

    % align the clustering outcome with ground truth
    res = bestMap(gnd, telabel);

    %% calculate the temp evaluation metric
    tmpACC(j) = length(find(gnd==res))/length(gnd);
    tmpNMI(j) = nmi(gnd, telabel);
    tmpARI(j) = clustereval(gnd, telabel, 'ari');
    tmpPurity(j) = Calculate_purity(gnd, telabel);
end
```

## Appendices

---

```
%% calculate the evaluation metric
ACC = mean(tmpACC);
NMI = mean(tmpNMI);
ARI = mean(tmpARI);
Purity = mean(tmpPurity);
end
```

### Appendix B. Matlab code for WAMKM

```
function [ACC, NMI, Purity, ARI, telabel, center] = mySPCL1(Xtr, Xte, gnd, opt)
%% example clear;clc; opt.sigma = 1:5; opt.k = 10; opt.i = 1; opt.spclType = 2;
%% [ACC, NMI, Purity, ARI, telabel, center] =
%% mySPCL1(rand(3000,50),rand(500,50),[ones(300,1);2*ones(200,1)],opt)
%% calculate the similarity matrix
W = EuDist2(Xtr,Xtr);

switch opt.para_tuning
    % parameter tuning
    case 1
        sigma=opt.sigma(opt.para_pos);
        % no parameter tuning
    case 0
        sigma = mean(mean(W));
end
% calculate the adjacent matrix
W = exp(-W.^2 ./ (2*sigma^2));
% calculate degree matrix
degs = sum(W, 2);
D = sparse(1:size(W, 1), 1:size(W, 2), degs);

% calculate weight matrix
weight=diag(D)/sum(diag(D));
ww = repmat(weight',size(W,1),1);
W=W.*ww;
% process the test data to match with the training data space
Xte = EuDist2(Xte,Xtr);
Xte = exp(-Xte.^2 ./ (2*sigma^2));
Xte = Xte.*repmat(weight',size(Xte,1),1);

for j=1:opt.kmeans_repeat
    %% run k-means clustering on weighted adjacent matrix directly
    [~,center] = kmeans(W, opt.k, 'start', 'cluster');
    % dist_all denotes distances of all data points in test data to clusters centres;
    dist_all = EuDist2(center,Xte);
    % select the minimum distance to a centre and assign class label to this centre;
    [~,telabel] = min(dist_all);
    telabel = telabel';

    % align the clustering outcome with ground truth
    res = bestMap(gnd,telabel);

    % calculate the temp evaluation metric
    tmpACC(j) = length(find(gnd==res))/length(gnd);
    tmpNMI(j) = nmi(gnd,telabel);
    tmpARI(j) = clustereval(gnd,telabel, 'ari');
```

## Appendices

---

```
tmpPurity(j)= Calculate_purity(gnd,telabel);

end
%% calculate the evaluation metric
ACC = mean(tmpACC);
NMI = mean(tmpNMI);
ARI = mean(tmpARI) ;
Purity = mean(tmpPurity);
end
```

### Appendix C. Matlab code for $k$ -means clustering

```
function []= Do_K_means_CV(dataset)
%% example: clear;clc;Do_K_means_CV(12)

%% tidy the output format
format short;
%% initialise parameters
iteration = 10; %repeat times
fold_k = 10; %  $k$  fold number
kmeans_repeat = 10; %  $k$ -means run times.

%% initialise matrix
ACC=zeros(iteration,fold_k);
Purity=zeros(iteration,fold_k);
NMI=zeros(iteration,fold_k);
ARI=zeros(iteration,fold_k);

%% dataset selection
switch dataset
case 1
    load 20news_uni_10fold.mat
case 2
    load binalpha_uni_10fold.mat
case 3
    load australian_uni_10fold.mat
case 4
    load Website_Phishing.mat
case 5
    load Contraceptive_Method_Choice.mat
case 6
    load diabetes_uni_10fold.mat
case 7
    load Coil20Data_25_uni_10fold.mat
case 8
    load Cardiotocography.mat
case 9
    load Spambase_10fold.mat
case 10
    load Parkinson_Speech_Dataset_10fold.mat
case 11
    load Solar_Flare_data2.mat
case 12
    load German_Credit_Data.mat
end
```

```

for ite = 1:iteration

    tic
    for i = 1:fold_k

        test = ind(:,ite) == i;
        train = ~test;
        trdata = Data(train,:);
        tedata = Data(test,:);
        trgnd = Y(train,:);
        tegnd = Y(test,:);

        %% data training process
        for j=1:kmeans_repeat
            [~,center] = kmeans(trdata, length(unique(trgnd)), 'start', 'cluster');
            % dist_all denotes distances of all data points in test data to clusters centres;
            dist_all = EuDist2(center,tedata);
            % testing process
            % select the minimum distance to a centre and assign class label to this centre;
            [~,telabel] = min(dist_all);
            telabel = telabel';
            res = bestMap(tegnd,telabel);
            tmpACC(j) = length(find(tegnd==res))/length(tegnd);
            tmpNMI(j) = nmi(tegnd,telabel);
            tmpARI(j) = clustereval(tegnd,telabel, 'ari');
            tmpPurity(j) = Calculate_purity(tegnd,telabel);

        end

        ACC(ite,i) = mean(tmpACC);
        NMI(ite,i) = mean(tmpNMI);
        ARI(ite,i) = mean(tmpARI);
        Purity(ite,i) = mean(tmpPurity);

    end

    toc

end

currenttime = datestr(now);
currenttime(currenttime==':')='.';
save(['./Results_Kmeans/Result_kmeans-',currenttime,'-',num2str(dataset)], 'ACC', 'NMI', 'Purity', 'ARI')

%% for a glance of the results.
mACC=mean(ACC(:))
sACC=std(ACC(:))

```



## Appendix D. Matlab code for performance evaluation

```
function [] = Do_SC_cv(dataset,method,para_tuning)
%% example: clear;clc;Do_mySC_cv(12,1,0)
%% % change opt.method to choose algorithm

% tidy the output format
format short;

iteration = 10; % repeat times
fold_k = 10; % k fold number
%% 0-spectral clustering; 1-weighted W; 2-k-means on W; 3-k-means on L
opt.method=method;
opt.cvk=5;
opt.cvind=[];
opt.kmeans_repeat=3;
opt.para_tuning = para_tuning;%% % para_tuning: 1: tuning, 0: no_tuning
opt.i=0;
opt.splType=2;%% % set clustering algorithm, 1=spectral clustering; 2=k-means clustering

%% initialise martix
ACC=zeros(iteration,fold_k);
Purity=zeros(iteration,fold_k);
NMI=zeros(iteration,fold_k);
ARI=zeros(iteration,fold_k);
best_para=zeros(iteration,fold_k);
%% initialise sigma
opt.sigma = 1:5;

switch dataset
case 1
    load 20news_uni_10fold.mat
case 2
    load binalpha_uni_10fold.mat
case 3
    load australian_uni_10fold.mat
case 4
    load Website_Phishing.mat
case 5
    load Contraceptive_Method_Choice.mat
case 6
    load diabetes_uni_10fold.mat
case 7
    load Coil20Data_25_uni_10fold.mat
case 8
    load Cardiotocography.mat
case 9
    load Spambase_10fold.mat
case 10
    load Parkinson_Speech_Dataset_10fold.mat
case 11
    load Solar_Flare_data2.mat
case 12
    load German_Credit_Data.mat
end

for ite = 1:iteration
```

```

tic
for i = 1:fold_k

    test = ind(:,ite) == i;
    train = ~test;
    trdata = Data(train,:);
    tedata = Data(test,:);
    trgnd = Y(train,:);
    tegnd = Y(test,:);

    [best_para(ite,i),ACC(ite,i),NMI(ite,i), Purity(ite,i),ARI(ite,i)]...
        = CV_train_cv1(trdata,trgnd,tedata,tegnd,opt);

end
toc
end

currenttime = datestr(now);
currenttime(currenttime==':')='.';
save(['./Results/LRSR_y1-' ,currenttime,'-',num2str(dataset),'-',num2str(method),'-
',num2str(para_tuning)],...
    'ACC','NMI','Purity','ARI','best_para','method','para_tuning')

%% for a glance of the results.
mACC=mean(ACC(:))
sACC=std(ACC(:))

function [best_para,ACC,NMI,Purity,ARI] = CV_train_cv1(trdata,trgnd,tedata,tegnd,opt)

switch opt.para_tuning
case 0 % no tuning
    num_cluster = unique(trgnd);
    opt.nClass=length(num_cluster);
    opt.k=opt.nClass;
    best_para=0;
case 1 % tuning
    num_cluster = unique(trgnd);
    opt.nClass=length(num_cluster);
    opt.k=opt.nClass;

    temp_pos = [];
    tempcvind=[];
    opt.cvind = zeros(size(trdata,1),1);

    for z = 1:length(num_cluster)
        temp_pos = find(trgnd == num_cluster(z));
        tempcvind = crossvalind('Kfold',length(temp_pos),opt.cvk);
        opt.cvind(temp_pos) = tempcvind;
        temp_pos = [];
        tempcvind=[];
    end

    ACC=zeros(length(opt.sigma),opt.cvk);
    for a=1:length(opt.sigma)
        opt.para_pos=a;
        for b = 1:opt.cvk
            test = opt.cvind == b;

```

```
train = ~test;

Xte=trdata(test,:);
Xtr=trdata(train,:);
Xgnd=trgnd(test);
switch opt.method
case 0
    ACC(a,b) = spcl(Xtr,Xte,Xgnd,opt);
case 1
    ACC(a,b) = mySPCL1(Xtr,Xte,Xgnd,opt);
case 2
    ACC(a,b) = mySPCL2(Xtr,Xte,Xgnd,opt);
case 3
    ACC(a,b) = mySPCL3(trdata,tedata,trgnd,opt);
end

end

end

%% best parameter combination
[~, ind] = max(ACC(:));
[m,~] = ind2sub(size(ACC),ind);
best_para=opt.sigma(m);
opt.para_pos = m;

end

switch opt.method
case 0
    [ACC, NMI, Purity, ARI] = spcl(trdata,tedata,tegnd,opt);
case 1
    [ACC, NMI, Purity, ARI] = mySPCL1(trdata,tedata,tegnd,opt);
case 2
    [ACC, NMI, Purity, ARI] = mySPCL2(trdata,tedata,tegnd,opt);
case 3
    [ACC, NMI, Purity, ARI] = mySPCL3(trdata,tedata,trgnd,opt);
end
```

## References

1. Jordan, M.I. and T.M. Mitchell, *Machine learning: Trends, perspectives, and prospects*. Science, 2015. **349**(6245): p. 255-260.
2. Hastie, T., R. Tibshirani, and J. Friedman, *Unsupervised Learning*, in *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, T. Hastie, R. Tibshirani, and J. Friedman, Editors. 2009, Springer New York: New York, NY. p. 485-585.
3. Liu, Q. and Y. Wu, *Supervised learning*. Encyclopedia of the Sciences of Learning, 2012: p. 3243-3245.
4. Xue, H., Q. Yang, and S. Chen, *SVM: Support vector machines*. 2009: Chapman & Hall/CRC: London, UK.
5. Safavian, S.R. and D. Landgrebe, *A survey of decision tree classifier methodology*. IEEE transactions on systems, man, and cybernetics, 1991. **21**(3): p. 660-674.
6. Ogutu, J.O., T. Schulz-Streeck, and H.-P. Piepho, *Genomic selection using regularized linear regression models: ridge regression, lasso, elastic net and their extensions*, in *BMC*. 2012. p. 34-41.
7. Chapelle, O., B. Scholkopf, and E. A. Zien, *Semi-Supervised Learning (Chapelle, O. et al., Eds.; 2006) [Book reviews]*. IEEE Transactions on Neural Networks, 2009. **20**(3): p. 542-542.
8. Prakash, V.J. and D.L. Nithya *A survey on semi-supervised learning techniques*. arXiv preprint arXiv:1402.4645, 2014. DOI: 10.14445/22312803/IJCTT-V8P105.
9. Zhu, X. and A.B. Goldberg, *Introduction to semi-supervised learning*. Synthesis lectures on artificial intelligence and machine learning, 2009. **3**(1): p. 1-130.
10. Cong, Y., J. Yuan, and J. Liu, *Sparse reconstruction cost for abnormal event detection*, in *CVPR*. 2011. p. 3449-3456.
11. Mehran, R., A. Oyama, and M. Shah, *Abnormal crowd behavior detection using social force model*, in *CVPR*. 2009, IEEE. p. 935-942.
12. Belkin, M. and P. Niyogi, *Laplacian eigenmaps for dimensionality reduction and data representation*. Neural computation, 2003. **15**(6): p. 1373-1396.
13. Dash, M., H. Liu, and J. Yao, *Dimensionality reduction of unsupervised data*, in *Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence*. 1997, IEEE. p. 532-539.
14. Lei, C. and X. Zhu, *Unsupervised feature selection via local structure learning and sparse learning*. Multimedia Tools and Applications, 2018. **77**(22): p. 29605-29622.
15. Kodinariya, T.M. and P.R. Makwana, *Review on determining number of Cluster in K-Means Clustering*. International Journal, 2013. **1**(6): p. 90-95.
16. Tibshirani, R., G. Walther, and T. Hastie, *Estimating the number of clusters in a data set via the gap statistic*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 06 January 2002. **63**(2): p. 411-423.
17. Kodinariya, T.M. and P.R. Makwana, *Review on determining number of Cluster in K-Means Clustering*. International Journal of Advance Research in Computer Science and Management Studies, 2013. **1**(6): p. 90-95.
18. Zheng, W., et al. *Unsupervised feature selection by self-paced learning regularization*. Pattern Recognition Letters, 2018. DOI: 10.1016/j.patrec.2018.06.029.
19. Abe, S., *Feature Selection and Extraction*, in *Support Vector Machines for Pattern Classification [Book]*. 2010, Springer London. p. 331-341.
20. Zhu, X., et al., *Graph PCA hashing for similarity search*. IEEE Transactions on Multimedia, 11 May 2017. **19**(9): p. 2033-2044.

## References

21. Zhu, X., et al. *One-step Multi-view Spectral Clustering*. IEEE Transactions on Knowledge and Data Engineering, 2018. DOI: 10.1109/TKDE.2018.2873378.
22. Zhang, S. *Cost-Sensitive KNN Classification*. Neurocomputing, 2019. DOI: 10.1016/j.neucom.2018.11.101.
23. Zhu, X., X. Li, and S. Zhang, *Block-row sparse multiview multilabel learning for image classification*. IEEE transactions on cybernetics, 2015. **46**(2): p. 450-461.
24. Arora, P. and S. Varshney, *Analysis of k-means and k-medoids algorithm for big data*. Procedia Computer Science, 2016. **78**: p. 507-512.
25. Bachem, O., et al., *Approximate K-Means++ in Sublinear Time*, in AAAI. 2016: Phoenix, Arizona USA. p. 1459-1467.
26. Malinen, M.I. and P. Fränti, *Balanced K-Means for Clustering*, in S+SSPR. 2014: Berlin, Heidelberg. p. 32-41.
27. Capó, M., A. Pérez, and J.A. Lozano, *An efficient approximation to the K-means clustering for massive data*. Knowledge-Based Systems, 2017. **117**: p. 56-69.
28. Birch, Z.T., *BIRCH: an efficient data clustering method for very large databases*, in SIGMOD, R.R. T. Zhang, M. Livny, Editor. 1996: New York. p. 103-114.
29. Guha, S., R. Rastogi, and K. Shim, *CURE: an efficient clustering algorithm for large databases*, in SIGMOD. 1998, ACM: Seattle, Washington, USA. p. 73-84.
30. Guha, S., Rastogi, R., & Shim, K., *ROCK: A robust clustering algorithm for categorical attributes*. Information Systems, 2000. **25**(5): p. 345-366.
31. Murtagh, F. and P. Contreras, *Algorithms for hierarchical clustering: an overview*. Wiley Data Mining and Knowledge Discovery, 2012. **2**(1): p. 86-97.
32. Lattanzi, S., et al., *Robust Hierarchical k-Center Clustering*, in ITCS. 2015, ACM: Rehovot, Israel. p. 211-218.
33. Gan, J. and Y. Tao, *DBSCAN revisited: mis-claim, un-fixability, and approximation*, in SIGMOD. 2015: Melbourne. p. 519-530.
34. Deng, Z., et al., *A scalable and fast OPTICS for clustering trajectory big data*. Cluster Computing, 2015. **18**(2): p. 549-562.
35. Cassisi, C., et al., *Enhancing density-based clustering: Parameter reduction and outlier detection*. Information Systems, 2013. **38**(3): p. 317-330.
36. Lv, Y., et al., *An efficient and scalable density-based clustering algorithm for datasets with complex structures*. Neurocomputing, 2016. **171**: p. 9-22.
37. Bryant, A. and K. Cios, *RNN-DBSCAN: A Density-Based Clustering Algorithm Using Reverse Nearest Neighbor Density Estimates*. IEEE Transactions on Knowledge and Data Engineering, 2018. **30**(6): p. 1109-1121.
38. Sharma, A. and A. Sharma, *KNN-DBSCAN: Using k-nearest neighbor information for parameter-free density based clustering*, in ICICICT. 2017: Kannur, India. p. 787-792.
39. Zheng, W., et al., *Dynamic graph learning for spectral feature selection*. Multimedia Tools and Applications, 2017. **77**(22): p. 29739-29755.
40. Zhang, S. *Multiple-scale cost sensitive decision tree learning*. World Wide Web, 2018. **21**, 1787-1800 DOI: 10.1007/s11280-018-0619-5.
41. Xu, D. and Y. Tian, *A comprehensive survey of clustering algorithms*. Annals of Data Science, 2015. **2**(2): p. 165-193.
42. Xu, X., et al., *Scan: a structural clustering algorithm for networks*, in KDD. 2007, ACM: San Jose, CA. p. 824-833.
43. Shiokawa, H., Y. Fujiwara, and M. Onizuka, *SCAN++: efficient algorithm for finding clusters, hubs and outliers on large-scale graphs*. Proceedings of the VLDB Endowment, 2015. **8**(11): p. 1178-1189.
44. Chang, L., et al., *Fast and Exact Structural Graph Clustering*. IEEE Transactions on Knowledge and Data Engineering, 2017. **29**(2): p. 387-401.
45. Shiokawa, H., T. Takahashi, and H. Kitagawa, *ScaleSCAN: Scalable Density-Based Graph Clustering*, in DEXA. 2018: Cham. p. 18-34.

## References

---

46. Zhu, X., et al. *Low-rank Sparse Subspace for Spectral Clustering*. IEEE Transactions on Knowledge and Data Engineering, 2018. DOI: 10.1109/TKDE.2018.2858782.
47. He, L., et al., *Fast Large-Scale Spectral Clustering via Explicit Feature Mapping*. IEEE Transactions on Cybernetics, 2018. **49**(3): p. 1058-1071.
48. STING, W.W.Y.J.M.R., *A Statistical Information Grid Approach to Spatial Data Mining*, in VLDB. 1997: Athens, Greece. p. 186-195.
49. Chen, J., et al., *FGCH: a fast and grid based clustering algorithm for hybrid data stream*. Applied Intelligence, 2018. **49**(4): p. 1228–1244.
50. Liu, F., C. Ye, and E. Zhu *Accurate Grid-based Clustering Algorithm with Diagonal Grid Searching and Merging*. in ICAMMT, 2017. DOI: 10.1088/1757-899X/242/1/012123.
51. Ding, Y. and X. Fu, *Kernel-based fuzzy c-means clustering algorithm based on genetic algorithm*. Neurocomputing, 2016. **188**: p. 233-238.
52. 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 Recognition, 2016. **51**: p. 310-321.
53. Du, L., et al., *Robust Multiple Kernel K-means Using L21-Norm*, in IJCAI. 2015: Buenos Aires, Argentina. p. 3476-3482.
54. Tibshirani, R., G. Walther, and T. Hastie, *Estimating the number of clusters in a data set via the gap statistic*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 2001. **63**(2): p. 411-423.
55. Jothi, R., S.K. Mohanty, and A. Ojha *DK-means: a deterministic K-means clustering algorithm for gene expression analysis*. Pattern Analysis and Applications, 2017. DOI: 10.1007/s10044-017-0673-0.
56. Zahra, S., et al., *Novel centroid selection approaches for KMeans-clustering based recommender systems*. Information sciences, 2015. **320**: p. 156-189.
57. Pavan, K.K., A.D. Rao, and G. Sridhar, *Single pass seed selection algorithm for k-means*. Computer Science 2010. **6**(1): p. 60-66.
58. Tremblay, N., et al., *Compressive spectral clustering*, in ICML. 2016: New York. p. 1002-1011.
59. Ng, A.Y., M.I. Jordan, and Y. Weiss, *On spectral clustering: Analysis and an algorithm*, in NIPS. 2002. p. 849-856.
60. Von Luxburg, U., *A tutorial on spectral clustering*. Statistics and computing, 2007. **17**(4): p. 395-416.
61. Jayasumana, S., et al., *Kernel methods on Riemannian manifolds with Gaussian RBF kernels*. IEEE transactions on pattern analysis and machine intelligence, 2015. **37**(12): p. 2464-2477.
62. Gebru, I.D., et al., *EM algorithms for weighted-data clustering with application to audio-visual scene analysis*. IEEE transactions on pattern analysis and machine intelligence, 2016. **38**(12): p. 2402-2415.
63. Zhu, X., H.-I. Suk, and D. Shen, *Low-rank dimensionality reduction for multi-modality neurodegenerative disease identification*. World Wide Web, 2019. **22**(2): p. 907–925.
64. Souza, C.R., *Kernel functions for machine learning applications*. Creative Commons Attribution-Noncommercial-Share Alike, 2010. **3**: p. 29-41.
65. Du, T., et al., *Spectral clustering algorithm combining local covariance matrix with normalization*. Neural Computing and Applications, 2018: p. 1-8.
66. Domeniconi, C. and M. Al-Razgan, *Weighted cluster ensembles: Methods and analysis*. ACM Transactions on Knowledge Discovery from Data (TKDD), 2009. **2**(4): p. 17-57.