Fast and Automatic Subspace Clustering Applying Modern Clustering Methods to the Subspace Clustering Problem

Yan Zhu

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Intelligent Systems)

Supervisor: Mimi Zhang, Joshua Tobin

August 2022

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Yan Zhu

August 19, 2022

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Yan Zhu

August 19, 2022

Fast and Automatic Subspace Clustering Applying Modern Clustering Methods to the Subspace Clustering Problem

Yan Zhu, Master of Science in Computer Science

University of Dublin, Trinity College, 2022

Supervisor: Mimi Zhang, Joshua Tobin

Due to the continuous development of big data technologies, researchers have an increasing demand for data analysis of high-dimensional data. Clustering analysis, one of the critical data analysis techniques, has many applications. For example, in bioinformatics or NLP, input datasets include tens to thousands of features (or dimensions), which makes data processing quite difficult. Clustering methods can help researchers to extract data with critical features from large data sets. Most modern subspace clustering methods apply spectral clustering algorithms for the final clustering step, which needs the exact number of clusters in advance. This thesis aims to optimise the final segmentation step to improve the algorithm's stability and obtain a method with good performance and good clustering results suitable for clustering analysis of high-dimensional data.

Acknowledgments

I would like to express my gratitude to my supervisors, Dr.Mimi Zhang and Joshua Tobin, for guiding me through this dissertation work. They have been involved in every step of my dissertation work, patiently answered all my questions and encouraged me.
I would like to thank my family for giving me a considerable amount of financial and emotional support.

Yan Zhu

University of Dublin, Trinity College
August 2022

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Clustering is a popular technique for the unsupervised classification of objects into clusters and allows for partitioning groups with similar features without labels. The clustering algorithm is essential to data analysis and other fields, with a wide range of applications. For example, in bioinformatics or NLP, the input datasets include tens to thousands of features (or dimensions), which presents considerable difficulties for data processing. The clustering methods can help researchers extract key data features from high-dimensional datasets.

With the continuous development of clustering technology, many excellent clustering algorithms have emerged. One of the well-known clustering techniques is the KMeans approach, which can be used to divide the input dataset into k clusters. While MacQueen (1967) coined the phrase, Stuart Lloyd of Bell Labs originally suggested the standard approach as a pulse-code modulation technique in 1957. However, due to the selection of the k value, the pure k-means method is not highly adaptable in real-world applications. The hierarchical clustering method is a well-known technique for cluster analysis and requires no specific number for the clusters. The output of hierarchical clustering is a tree diagram structure, which provides a convenient way to explore all levels of entity relationships (Ma and Dhavala (2018)). By leveraging the spectrum (eigenvalues) of the similarity matrix, the spectral clustering technique minimizes the number of dimensions before clustering the data. The normalized cuts algorithm and Shi-Malik algorithm are popular normalized spectral clustering methods proposed by Shi and Malik (2000), which are mostly utilized in the study of image segmentation. In density-based clustering, clusters are defined as denser regions than the rest of the dataset and sparse regions are delineated by defining noise and boundary points (Kriegel et al. (2011)). The most popular density-based clustering method is currently DBSCAN (Ester et al. (1996)). In

contrast to many unique methods, it has a clearly defined cluster model dubbed "density-accessibility," which is comparable to link-based clustering in that it is based on connection points under a particular distance threshold.

With the continuous development of big data technology, data collection handled by data analysis is becoming increasingly significant, which in part makes traditional clustering algorithms technically difficult in clustering some specific data set. For example, in the high-dimensional sparse data, the cluster class only exist in the subspace composed of some attributes. Some valuable features are embedded in many features, and traditional clustering algorithms are less efficient in obtaining subspace clusters. Suppose a clustering algorithm can identify the subspace cluster and directly find the essential features among a huge amount of data points. In that case, it will play a vital role in establishing a good clustering or classification model. The subspace clustering method is a clustering algorithm using the advanced step of feature selection based on the feature correlation to detect all clusters in all subspaces. However, the subspace clustering methods seldom pay attention to the final segmentation step. Almost all the subspace clustering methods use spectral clustering as the last step to finding the clusters. However, the spectral clustering method is suitable for datasets known to have a specific number of clusters. The experimental results of spectral clustering methods are inconsistent from run to run, and the execution time for a massive amount of data points is slow. In this paper, we mainly focus on optimizing the last part of the segmentation using a density-based method (DBSCAN) instead of spectral clustering as the final step to improve the stability of the algorithm and obtain a method suitable for high-dimensional data clustering analysis with good performance and clustering effect.

1.2 Contributions

In this paper, we have completed the following contributions:

(1) Reviewed the traditional and subspace clustering methods, and focus particularly on spectral-based subspace clustering methods.

(2) Learned about and implemented KMeans clustering methods as baseline, Sparse Subspace Clustering, Sparse subspace clustering by orthogonal matching pursuit (SSC-OMP), Elastic net Subspace Clustering (EnSC), Low-Rank Representation (LRR), and Least squares subspace clustering (LSSC).

(3) Generated synthetic data on different angles, noise levels and dimensions. Applied clustering methods on synthetic data to analyse the clustering performance of the different clustering methods.

(3) Proposed a novel method to improve the clustering performance of subspace clustering method by replacing spectral clustering with DBSCAN on the final clustering step.

(4) Experiment the novel approach using real data Hopkins 155 dataset and Extended Yale B database.

1.3 Thesis Structure

The overall paper is organized as follows. The literature review of various clustering methods will be discussed in *Chapter 2*. *Chapter 3* presents the experiment of popular existing sparse Subspace Clustering methods on the synthetic data of varying angles, noise level and dimensions. In *Chapter 4*, we describe a novel method using DBSCAN instead of spectral clustering with sparse representation. And finally, we discuss the results of the experiment of the novel method on real data and conclusion in *Chapter 5*. The objective of this paper is to propose a novel clustering method, apply it into real data and evaluate its performance.

Chapter 2

Literature Review

2.1 Clustering Methods

Numerous studies have been done on supervised classification, and neural networks, decision trees, Bayesian classifiers, etc., all contain some of the first supervised classification techniques. Unsupervised classification differs from supervised classification, where we are given tagged patterns. In contrast to supervised classification, which provides us with labelled patterns, the unsupervised classification does not apply any labels to any patterns. Clustering is a common name for unsupervised categorization. In machine learning, data mining, and bioinformatics, clustering is a commonly used statistical data analysis approach. As an initial step to comprehending the topics (datasets) in a machine learning system, we frequently group instances together in machine learning. Clustering is the process of putting unlabeled instances together and is based on unsupervised machine learning because the samples are unlabeled. When examples include labels, clustering turns into classification. In general, given a set of subsets $S_1, S_2, \cdots, S_k$, the step of the clustering structure can be represented as:

$$S_1 \cap S_2 \cap S_3, \cdots, \cap S_k = \emptyset \tag{2.1}$$

, which means that each element in $S(S_1, \cdots, S_k)$ will be a member of the specified subset and not the overlap subset. Because no patterns are labelled in clustering, it is thought to be more challenging than supervised classification. When using supervised classification, the assigned labels serve as a guide for classifying the individual data objects. It might be challenging to determine which category a pattern will belong to without labelling. Several variables or traits may be thought to be good candidates for clustering. The issue can get worse due to the curse of dimensionality. High dimensionality affects the algorithm's consistency and increases processing costs. Although there are some features,

there have been reported solutions in the form of selection methods (Saxena et al. (2010)).

Hierarchical clustering and partitional clustering are two types of data clustering techniques. While partitional clustering algorithms select all clusters at once, hierarchical clustering techniques leverage previously discovered clusters to find subsequent clusters. Hierarchical methods can be unifying (bottom-up) or dividing (top-down). Each component begins as a separate cluster in an agglomerative algorithm, which subsequently combines them into a single, larger cluster. Division algorithm starts with the complete collection and then splits it into more manageable clusters (Madhulatha (2012)).

Partitional clustering minimises a given criterion of clustering by iteratively repositioning data points between clusters until the best division is obtained. Algorithms for partitioned clustering separate the data points into k partitions, each representing a cluster. Partitioning is done by using an objective function. In order to make things within clusters "similar" and objects in other clusters "dissimilar," clusters are built to maximize objective partitioning criteria, such as a distance-based dissimilarity function. Applications that demand a specific number of clusters can benefit from partitioned clustering techniques (Popat and Emmanuel (2014)). Partitional methods are advantageous for applications needing large data sets for which the production of a dendrogram is computationally prohibitive. The partitional methods generate clusters by optimizing a function to define the local or global criterion. Here are some of the popular clustering methods (Jain et al. (1999)):

a. *Squared Error Clustering Method*:

   (1) Choose the first set of patterns that are divided into a defined number of clusters and cluster centres.

   (2) Assign each pattern to the closest cluster centre, then use that location to determine the cluster's centroid. Repeat this process until the cluster members are stable or until convergence is obtained.

   (3) Merge and divide clusters depending on heuristic data, using step 2 repetition as an option

b. *Graph-Theoretic Clustering*, which is based on the concepts of the *minimal spanning tree*, aslo known as MST of the data. However, the MST edges in the largest dataset need to be deleted to generate cluster efficiently.

Hierarchical clustering algorithms divide or merge data sets into a series of nested partitions (Murtagh and Contreras (2017)). Clustering in the cohesive approach begins with each object in a distinct cluster and then moves on to cluster the closest cluster pairings until all objects are gathered in a single cluster. Contrarily, in split hierarchical

clustering, all items are initially grouped into a single cluster before being divided into smaller clusters until they are all grouped into unitary clusters (Voorhees (1986), Rani[1] and Rohil (2013)). All of these hierarchical methods show a natural way of representing clusters, called a tree diagram. ROCK, BIRCH (which uses hierarchical balanced iterative reduction and clustering), and CURE are examples of such algorithms (using representative clustering) (Popat and Emmanuel (2014)). Sibson (1973), Rohlf (1973) and Defays (1977) proposed the efficient hierarchical clustering algorithms with $O(n^2)$ time complexity of the single link methods and a nonunique complete link algorithm , which have been widely used. There have been numerous grouping and stratification techniques put out through time. These techniques can use graphical representations, such as Sneath and Sokal (1966) created for building hierarchical clusters. The second category of hierarchical clustering techniques includes centroid, median, and minimal variance methods, enabling cluster centres to be selected. The second one canbe described by the dissimilaries or alternatively by the cluster center, coordinates (Murtagh and Contreras (2012), Murtagh and Contreras (2017)).

2.2 Density-Based Clustering Methods

Hierarchical clustering is a clustering algorithm gathering data points whit same features and dividing data points with opposite features. Hierarchical clustering has two different approaches: Agglomerative clustering and Divisive clustering. Agglomerative hierarchical clustering methods represent a cluster with pixel description and utilize the nest sequence to combine the clusters. In comparison, divisive hierarchical clustering will consider all the patterns in one of the clusters and divide them then. And the steps of the hierarchical clustering methods are easily interpreted by using the graph representation and can ignore the effect of the initial set-up and local minima Wilson et al. (2002) Frigui and Krishnapuram (1997). The partitional clustering algorithms, which employ the sum of distances from the vector of the pixel to the prototype of the cluster contained in n-dimension space, will, however, output a single partition of the data via iterations. And each pixel is set from 0 to 1 as an extent to each cluster assigned. And the process of the partitional clustering algorithms combine with the wide range of knowledge from different aspects by utilizing various objective distance functions Wilson et al. (2002) Frigui and Krishnapuram (1997). The most significant difference between hierarchical and partial clustering is the running time. The Partitional algorithm processes a piece of data and finds the best point for that piece of data, and it is faster than the hierarchical approach. However, the higher the value of k, the higher the transaction cost. Another difference is the input-output. This difference consists of the different methods of layered and partial

algorithms. A hierarchical algorithm finds many optimality points, usually giving us a local optimum rather than a global one and sometimes, this local optimum may be the global optimum. But hierarchical methods handle all of the data points and also provide an optimum point to handle it.

Mean shift stands for a procedure of iteration that can shift each instance (data points) to the average data points around them and builds upon the concpet oif the kernel density estimation. Given a set of data S embedded in a n-dimensional space of Euclidean and a *flat kernel* K, which is the obejective function in the $\lambda - ball$:

$$K(x) = \begin{cases} 1 & if \|x\| \leq \lambda \\ 0 & if \|x\| > \lambda \end{cases} \tag{2.2}$$

And given $x \in X$, the *sample mean* is defined as:

$$m(x) = \frac{\sum_{x \in S} K(s-x)s}{\sum_{s \in S} K(s-x} \tag{2.3}$$

According to the above equations, Fukunaga and Hostetler (1975) called the difference $m(x) - x$ as *mean shift* and the repeated operations of the data points is the *mean shift algorithm*, which has been used for the cluster analysis (Cheng (1995)). Here are two main mean shift clustering algorithms (Anand et al. (2013)):

a. **Euclidean Mean Shift Clustering**: Given n data points embeded in d-dimensional space and the sample data points density estimator obtains with the kernel $k(x)$ defined by:

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h_i^d} k(\| \frac{x - x_i}{h_i} \|^2) \tag{2.4}$$

The local maximum of the objective function can be reached using mean shift process iteratively:

$$\delta x = \frac{\sum_{i=1}^{n} \frac{x_i}{h_i^{d+2}} g(\| \frac{x-x_i}{h_i} \|^2)}{\sum_{i=1}^{n} \frac{1}{h_i^{d+2}} g(\| \frac{x-x_i}{h_i} \|^2)} - x \tag{2.5}$$

, where the current mean is x and the δx is the mean shift vector.

b. **Mean Shift Clustering in Kernel Spaces**: Anand et al. (2013) improved the above mean shift clustering algorithm from the euclidean space to the general space. Given χ as the input space and data points $x_i \in \chi$. Each data point is mapped to a d_ϕ-dimensional feature space applying the following objective function:

$$\phi(x) = [\phi_1(x) \ \phi_2(x) \ \cdots \ \phi_{d_phi}(x)]^T \tag{2.6}$$

The mean shift procedure can be iterated:

$$\delta y = \frac{\sum_{i=1}^{n} \frac{\phi(x_i)}{h_i^{d_\phi+2}} g(\|\frac{y-\phi(x_i)}{h_i}\|^2)}{\sum_{i=1}^{n} \frac{1}{h_i^{d_\phi+2}} g(\|\frac{y-\phi(x_i)}{h_i}\|^2)} - y \tag{2.7}$$

Rodriguez and Laio (2014) proposed the **Density Peaks Clustering** method, which uses the local density ρ_i and its distance δ_i from the data points in the high density space. Given the dataset $X_{N\times M} = [x_1, x_2, \cdots, x_N]^T$ and $x_i = [x_{1i}, x_{2i}, x_{Mi}]$ is a vector with M attributes and the Euclidean distance $d(x_i, x_j)$ defined as follows:

$$d(x_i, x_j) = \|x_i - x_{j2}\| \tag{2.8}$$

The local density of a data points is defined as:

$$\rho_i = \sum_j \chi(d(x_i, x_j) - d_c) \tag{2.9}$$

$$\chi(x) = \begin{cases} 1, & x < 0 \\ 0, & x \leq 0 \end{cases} \tag{2.10}$$

ρ_i can also be defined as:

$$\rho_i = \sum_j \exp(-\frac{d(x_i, x_j)^2}{d_c^2}) \tag{2.11}$$

, where d_c is an input parameter, which controls the degradation rate of the weight. The calculation of δ_i contains the minimum distance and defined as:

$$\delta_i = \begin{cases} \min\limits_{j:\rho_i>\rho_j} (d(x_i, x_j)), & if\ \exists\ j \quad s.t. \rho_i > \rho_j \\ \max\limits_{j} (d(x_i, x_j)), & otherwise \end{cases} \tag{2.12}$$

When using a local cluster criterion, density-based clustering algorithms define clusters as areas in the data space with higher density than areas with noise or boundary points. However, the high-density zones may be arbitrarily separated into the data points, with the potential for arbitrary sizes and forms. The grid-based subspace clustering method is one of the common ways to find the high-density area, which means partitioning each dimension space into non-overlapping grids (Jahirabadkar and Kulkarni (2013)). Clusters in high-dimensional data are frequently and widely found using density-based methods. These algorithms find the subspaces in the high-density data, and the target clusters are contained within them.

The various density-based methods proposed in the literature mainly differ in the following aspects:

1. How to estimate the density $p(x)$?
2. How to define the connectivity?
3. Is the algorithm suitable for achieving scalability for a large dataset?

DBSCAN (Ester et al. (1996)) is the first density based clustering algorithm, which based on the concept of density. In DBSCAN, given a distance threshold eps and a density threshold number $minPts$:

1. eps : It represents that if the distance between two data points is lower or equal to eps, these points can be considered as neighbors.
2. $minPts$: It represents the minimum number of points to form a dense region.

Density-based spatial clustering of applications with noise (DBSCAN), a new clustering technique described by Ester et al., uses just one input parameter and assists the user in choosing a suitable value. The DBSCAN algorithm applies both 2D and 3D Euclidean space as well as some high density space and can formalize the clusters and noise intuitively. The main idea of this method is that for each point in the cluster, the density of a given radius domain must be greater than a certain threshold. The *Eps-neighborhood* of a point is defined:

$$N_{Eps(p)} = \{q \in \mathbb{D} | dist(p, q) \leq Eps\}$$

If a point p is *direct density-reachable* from a point q, that means:

1) p$\in N_{Eps(q)}$
2) $\left|N_{Eps}(q)\right| \geq MinPts$ (Core condition)

Algorithm 1 Density-based Spatial Clustering of Applications with Noise (DBSCAN)(Ester et al. (1996))

Input: Data matrix D, radius Eps and density threshold $MinPts$.

```
∀n ∈ 1 : N do
if n not member of cluster then
    Create new cluster
    while neighbors of n satisfy do
        add them to cluster
    end while
end if
```

DBSCAN clustering model is a density-based spatial clustering algorithm, which splits regions with adequate density into clusters and detects clusters of any shape in a noisy spatial database. According to the conventional definition of density, the number of points (including the point itself) that fall inside a circle centred on point a and with radius r can be used to estimate the density of point an in a data collection given a radius r. Density depends on radius. The points in the sample set can be divided into the following three groups using the definition of density provided above:

a. Core points: points that contain more than the number of MinPts (the minimum number) in the region of radius r are called core points.

b. Boundary points: points in the region of radius r that contain less than the number of MinPts but are direct neighbours of the core points.

c. Noise points: points that are neither core points nor boundary points. Noise points will not be included in the clusters, and boundary points and core points form the 'clusters' of the clusters.

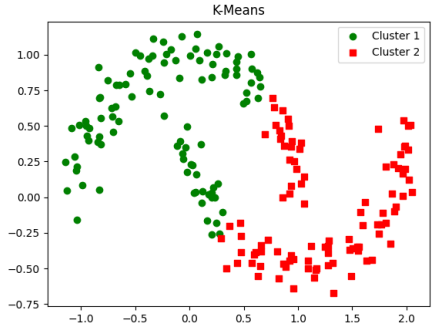

Figure 2.1: Clustering using the KMeans algorithm

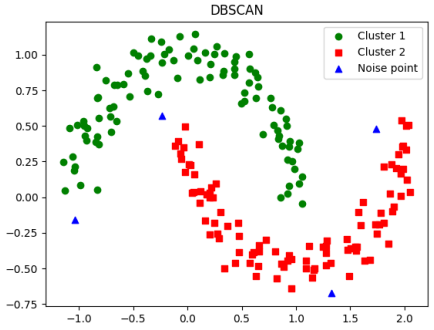

Figure 2.2: Clustering using the DBSCAN algorithm

The Figure 2.1 & 2.2 shows the clustering effects of KMeans algorithm and DBSCAN algorithm on two half-moon data sets. It can be seen from the figures that the DBSCAN algorithm can accurately identify two half-moon-shaped data clusters. Compared with the DBSCAN algorithm, the KMeans algorithm has lower accuracy.

The clustering model using DBSCAN utilizes a simple minimum estimation of the density level regarding to a threshold about the number of the neighbors. The objective of the DBSCAN clustering model is to divide the areas of them using the criteria of satisfying the minimum density. There are two important parameters in DBSCAN clustering model: the number of neighbors $minPts$ and the radius ε. All the points in the neighbor of a core point within the radius ε of it can be considered as the same cluster of the core point (also

called *direct density reachable*). If there is no core point in the set, they are called *border points* and all the data points in this set are *density connected* (Schubert et al. (2017)).

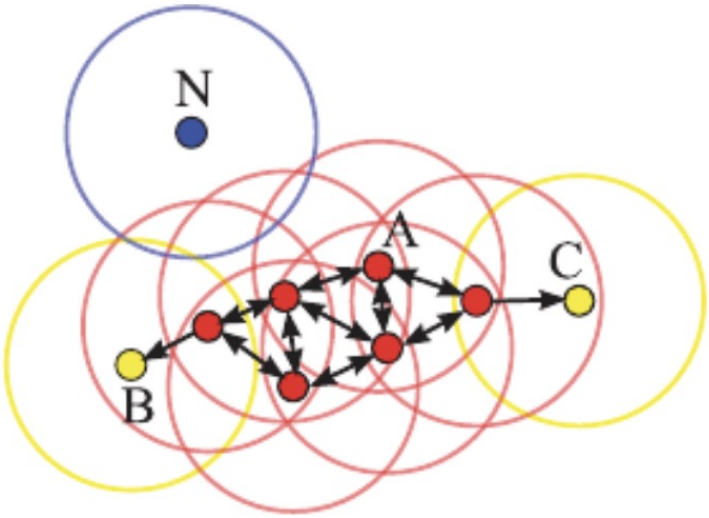

Figure 2.3: Illustration of the DBSCAN cluster model (Schubert et al. (2017))

The Figure 2.3 shows the concepts of the DBSCAN clustering model. In the situation of this figure,the parameter ***minPts*** is 4 and the radius of the core point is illustrated by the circles' radius. Point A is a *core point*, Point B and C are the *border points* while Point N is a *noise point*.

The DBSCAN algorithm linearly scans the set of data points to look for things that haven't been processed yet. Its neighbours are iteratively extended and added to the cluster when a core point is discovered. Non-core points are allocated to the noise. Objects given to a cluster are skipped when encountered in subsequent linear scans. This fundamental algorithm is the standard method for computing inverse closures of relationships with the minimal modification that only the core points are extended. Algorithm 2 gives simplified pseudocode of this DBSCAN algorithm.

Algorithm 2 Original Sequential DBSCAN Algorithm(Schubert et al. (2017))

Input: A set of database, radius ε, density threshold $minPts$, distance function $dist$ and the labels of the data points.

```
1: foreach point p in data set do
2:     if label(p) ≠ undefined then continue
3:     Neighbors N ← RangeQuery(DB, dist, p, ε)
4:     if |N| < minPts then
5:         label(p) ← Noise
6:         continue
7:     c ← next cluster label
8:     label(p) ← c
9:     Seed set S ← N\{p}
10:    foreach q in S do
11:        if label(q) = Noise then label(q) ← c
12:        if label(q) ≠ undefined then continue
13:        Neighbors N ← RangeQuery(DB, dist, q, ε)
14:        label(q) ← c
15:        if |N| <minPts then continue
16:        S ← S ∪ N
```

The original DBSCAN clustering model is a simplified pseudo-code based on the concept of DBSCAN model. There is a function called *RangeQuery*, which only execute if the label isn't labelled.

CLIQUE is the first grid-based, non-overlapping subspace clustering algorithm for the high dimensional data, which discovers the subspaces of them (Agrawal et al. (1998)). CLIQUE utilizes an Apriori-like method to recursively navigate the possible subspaces from the bottom to the top. An axis-parallel grid is used to partition the data space into equal-sized blocks, and only cells with densities higher than a certain threshold are kept. Finding such dense cells begins with one-dimensional dense cells using a bottom-up technique. An iterative process that goes from (k-1) dimensional dense cells to (k-2) dimensional dense cells develops k-dimensional cells by self-joining all of the (k-1) dimensional dense cells with the first (k-2) dimensions. All candidates that were generated but were not dense will be removed. The clusters are discovered to be the most extensive collection of connected dense cells after creating all the dense cells of interest.

SUBCLU is the first subspace clustering algorithm that extends the DBSCAN algorithm to detect clusters in high dimensional data based on the notion of DBSCAN (Kailing et al. (2004)). It discovers all the subspaces in the high dimension space using

a greedy algorithm to overcome the grid-based approaches' limitations. The input density parameters ε-radius and μ-density threshold are used to execute DBSCAN on each dimension to initially construct all 1-dimensional clusters. This method holds the monotonicity attribute in the core object specification that is used to define the clusters. This is used to locate clusters in all high-dimensional candidate subspaces using an Apriori-based method (Jahirabadkar and Kulkarni (2013)).

2.3 Spectral Clustering Method

Spectral clustering algorithm is a popular clustering method, which doesn't require the assumptions of the whole structure of the data set. Spectral clustering algorithms perform well for the data set with arbitrary size and shape. In 1973, Donath and Hoffman (2003) first proposed the concept of the spectral clustering model and introduced the technique of graph partitioning. At the same year, Fiedler (1973) gave the proof of the dichotomy of the graph is closely related to the second eigenvector of the Laplacian matrix. Hagen and Kahng (1992) discovered the relations between graph partition and clustering and proposed the ratio cut method first in 1992. Since 2000, spectral clustering model becomes a hot spot of many areas such as data mining, bio-information. computer vision and image classification (Jia et al. (2014)).

Von Luxburg (2007) described different graph Laplacian operators and their basic properties, deduces several common algorithms in detail through several different ways, and points out their advantages and disadvantages. Given a matrix W, the simple way is to solve the mincut problem:

$$cut(A_1, \cdots, A_k) := \frac{1}{2}\sum_{i=1}^{k} W(A_i, \bar{A}_i) \tag{2.13}$$

The mincut occasionally only isolates a single vertex from the rest of the graph, which is not what we want. One of the solution is to make the sets $A_1, \cdots, A_k$ relatively large. And the RatioCut and normalized cut Ncut is two most common solution to solve this problem:

$$RatioCut(A_1, \cdots, A_k) := \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{|A_i|} = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{|A_i|} \tag{2.14}$$

$$Ncut(A_1, \cdots, A_k) := \frac{1}{2}\sum_{i=1}^{k} \frac{W(A_i, \bar{A}_i)}{vol(A_i)} = \sum_{i=1}^{k} \frac{cut(A_i, \bar{A}_i)}{vol(A_i)} \tag{2.15}$$

The two objective functions discussed above thus aim to achieve a "balance" of clusters, as determined by the number of vertices or the weights of edges. But the formerly straight-

forward mincut problem becomes NP-hard with the addition of equilibrium requirements. This kind of issue can be facilitated by spectral clustering.

2.4 Subspace Clustering Methods

Clustering algorithms aim to find groupings of related things that are often represented as a point in a multi-dimensional space. Technology improvements have facilitated and accelerated data acquisition, leading to larger, more complex datasets with numerous objects and dimensions. Existing techniques must be modified to preserve the quality of clustering, and speed as datasets expand and change. When different dimensional subsets of the dataset are connected differently, multidimensionality still affects many older clustering algorithms. Algorithms for subspace clustering aim to reveal this kind of link. Unimportant attributes must be eliminated for the clustering algorithm to concentrate solely on the pertinent two-dimensional space to discover these clusters. In addition to being simpler to understand, the groups located in the lower dimensional area can better direct future research. The idea of 'Subspace Clustering' has been introduced to address the issue of the clusters being embedded in different subsets of dimensions in high-dimensional data.

To locate clusters in various subspaces of the same dataset, subspace clustering is an extension of feature selection. The number of axis-parallel subspaces where clustering may exist grows exponentially as the size of the data space rises. To develop suitable trial methods for locating subspaces, research in this area is focused mainly on this. Beginning with state-of-the-art methods for axis-parallel subspace clustering, two conflicting fundamental strategies for searching subspaces—top-down search and bottom-up search—are pursued. It is essential to use a heuristic approach to find suitable subspaces, which determines the subspace clustering algorithms' characteristics, which presents a difficulty for subspace clustering algorithms (Friedman (1994), Strang (2006)). Figure 2.4 presents a hierarchy of subspace clustering algorithm organized by the the search techniques.

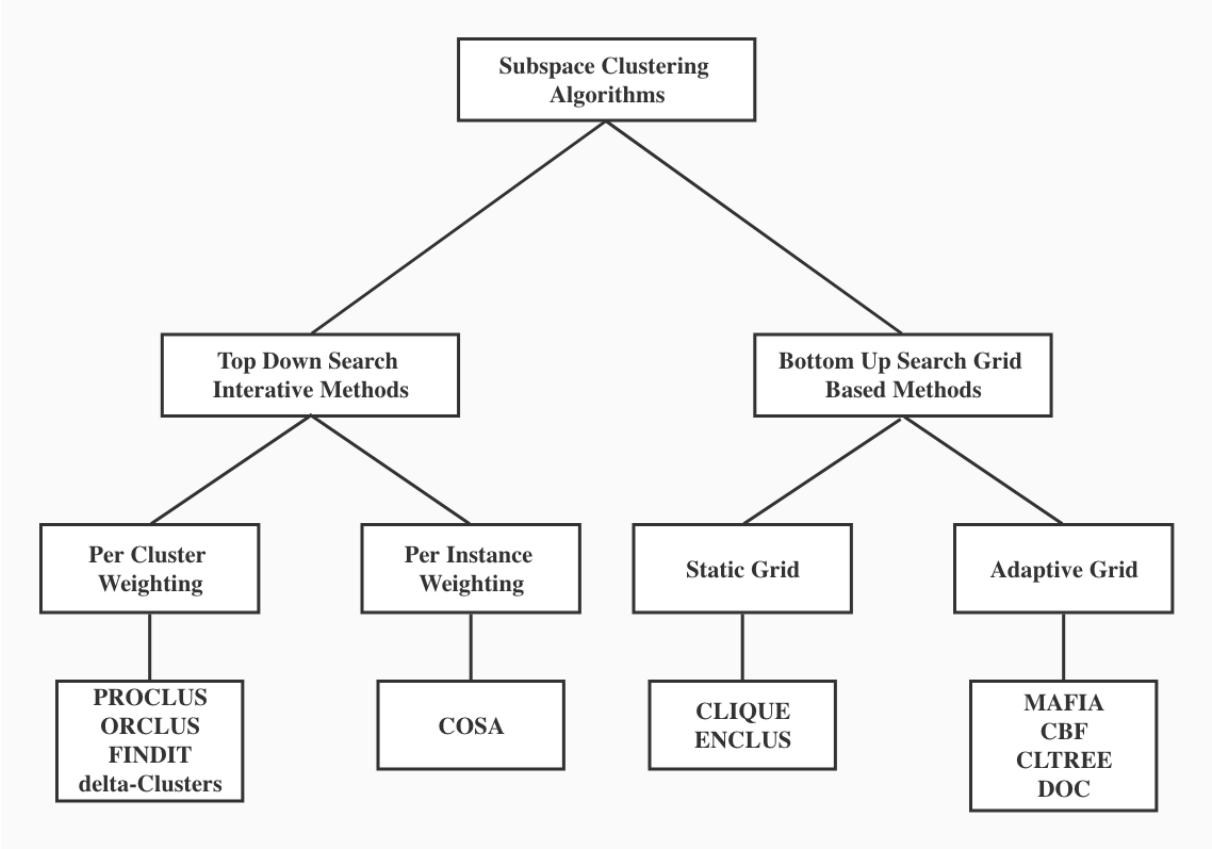

Figure 2.4: Hierarchy of Subspace Clustering Algorithms (Parsons et al. (2004)
)

Top-Down Subspace Search The top-down approach is justified by identifying subspaces of clusters starting from the complete dimensional space. This is often accomplished by selecting a subset of characteristics for a given collection of points (possible cluster members) such that these points match the specified clustering requirements when projected onto that particular subspace. The challenge is that at least some of the cluster members must be known to determine the subspace of a cluster. On the other hand, the subspace of each cluster must be understood to identify its members. Most top-down systems rely on the localization assumption, which is a somewhat rigid premise, to avoid this circular dependence. It is expected that the local neighbourhoods (in the full-dimensional data space) of cluster centroid that can be used to generate the subspaces of a cluster. In other words, even in the full-dimensional space, it is assumed that the subspaces of each cluster may be learned from the local neighbourhoods of cluster representatives or cluster members. Other top-down approaches that do not rely on locality build a collection of prospective cluster members using random sampling as a heuristic (Kriegel et al. (2012)).

Bottom-Up Subspace Search The search space of the frequent itemset issue in the transactional database domain of shopping basket analysis is similar to the exponential

search space of all feasible subspaces of the data space to be traversed. For example, an itemset may contain lots of items. The core concept behind the APRIORI algorithm (Agrawal et al. (1994)) is to start with a size 1 itemset (referred to as a "transaction") that contains a single item, regardless of any additional items that may be present, and then exclude from the search larger itemsets that are no longer frequent because it is already known which smaller itemsets are frequent (Kriegel et al. (2012)).

2.5 Sparse Subspace Clustering Methods

Sparse Subspace Clustering Since each data point in a subspace union can be effectively represented as a linear or affine combination of other points, Elhamifar and Vidal (2012) proposed and studied an algorithm based on this self-expressive property of the data. This algorithm, called sparse subspace clustering (SSC), is based on prior research on subspace clustering methods. The approach can solve issues with local spectrum-based clustering algorithms, including selecting the right neighbourhood size and handling points close to subspace intersections. By incorporating the necessary models into the sparse optimisation process, the sparse subspace clustering approach may directly address data disturbances like noise, sparse discrete entries and missing entries, and more broad classes of affine subspaces. The sparse subspace clustering successfully retrieves the precise sparse representation of the data set when the subspaces and data point distribution are appropriate.

Let $\{S_\ell\}_{\ell=1}^{n}$ be the representation of n linear subspace of dimensions and given a set of N non-noise data points $\{y_i\}_{i=1}^{N}$ which embedded in the union of n subspaces. The matrix containing all data points is represented as:

$$Y \triangleq [y_1 \cdots y_N] = [Y_1 \cdots Y_n]\Gamma \tag{2.16}$$

, where $Y_\ell \in \mathbb{R}^{D \times N_l}$ is a $rank - d_l$ matrix of the $N_l > d_l$ points that lie in S_l and $\Gamma \in \mathbb{R}^{N \times N}$ is an unknown permutation matrix. To solve the clustering problem in two steps. In the first step, we find several other points belonging to the same subspace for each data point. To do this, the authors propose a global sparse optimisation procedure whose solution encodes information about the membership of the data points in the underlying subspace of each point. Information about the data point's membership in each point's underlying subspace. In a second step, use this information in the spectral clustering framework to infer the clustering of the data. The following equation:

$$y_i = Yc_i, \quad c_{ii} = 0 \tag{2.17}$$

, which has infinite solutions and can be restricted solutions by minimizing some objective functions. For example, minimize the $\ell_q - norm$ solutions:

$$\min \|C_i\|_q \quad s.t. \quad y_i = Yc_i, c_{ii} = 0 \tag{2.18}$$

After solving the above optimizing problem, we will obtain sparse representations. The subsequent phase involves using the sparse coefficients to infer the segmentation of the data points into several subspaces. We can use a weighted graph $\mathcal{G} = (\nu, \varepsilon, W)$, where ν refers to the set of N nodes in the graph, $\varepsilon \subseteq \nu \times \nu$, and $W \in \mathbb{R}^{N \times N}$ is a symmetric matrix which represents the edges of the graph. The following Algorithm 1 summarizes the basic steps of the Sparse Subspace Clustering algorithm.

Algorithm 3 Sparse Subspace Clustering (SSC)(Elhamifar and Vidal (2012))

Input: Given a set of data points $\{y_i\}_{i=1}^N$ in the n linear subspaces $\{S_i\}_{i=1}^n$:

1: Use the following formula to solve the sparse optimization program:

2: For uncorrupted data:

$$\min \|C\|_1 \quad s.t. \quad Y = YC, \quad diag(C) = 0$$

3: For corrupted data:

$$\min \|C\|_1 \quad + \lambda_e \|\|_1 + \frac{\lambda_z}{2}\|Z\|_F^2$$
$$s.t. \quad Y = YC + E + Z, \quad diag(C) = 0$$

4: Normalize C as $c_i \leftarrow \frac{c_i}{\|c_i\|_\infty}$.

5: Set the weights on the edges of a similarity graph with N nodes:

$$W = |C| + |C|^T$$

6: Apply sparse subspace clustering method into the similarity graph built by n connected components embedded in the n subspace:

$$W = \begin{bmatrix} W_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & W_n \end{bmatrix} \Gamma$$

Output: Finally, output the segmentation of the data points:$Y_1, Y_2, \cdots, Y_n$.

Sparse Subspace Clustering By Orthogonal Matching Pursuit You et al. (2016b) studied a subspace clustering method based on orthogonal matching pursuit, and

proved that it was computationally effective, and guaranteed the similarity of subspace under a wide range of conditions. The sparse representation can be solved by following optimizing problem since every data points in S_i can be expressed:

$$C_j^* = \arg\min_{c_j} \|c_j\|_0 \quad s.t. \quad x_j = Xc_j, c_{jj} = 0 \tag{2.19}$$

, where $\|c\|_0$ is the number of the nonzero points in c. Elhamifar and Vidal optimized the above NP hard problem using following ℓ_1 solution:

$$C_j^* = \arg\min_{c_j} \|c_j\|_1 \quad s.t. \quad x_j = Xc_j, c_{jj} = 0 \tag{2.20}$$

The above two equations provide *subspace preserving* by providing a sparse representation c_j, and under certain conditions, the orthogonal matching pursuit algorithm (Algorithm 4) can solve the problem $\min_c \|Ac - b\|_2^2 \quad s.t. \quad \|c\|_0 \leq k$ by choosing one column of $A = [a_1, \cdots, a_M]$ and calculating the coefficients for the selected one until k elements are selected. The $c_j^* \in \mathbb{R}^N$ can be calculated by OMP while a zero is inserted into the jth entry. And then, using spectral clustering to the affinity matrix W as the Algorithm 5 shown.

Algorithm 4 Orthogonal Matching Pursuit (OMP)(You et al. (2016b))

Input: $A = [a_1, \cdots, a_M] \in \mathbb{R}^{m \times M}, k_{\max}, \epsilon$.

1: **Initialize** $k = 0$, residual $q_0 = b$, support set $T_0 \neq 0$.
2: **while** $k < k_{\max}$ and $\|q_k\|_2 > \epsilon$ **do**
3: $T_{k+1} = T_k \cup \{i^*\}$, where $i^* = \arg_{i=1,\cdots,M} \max |a_i^T q_k|^1$.
4: $q_{k+1} = (I - P_{T_{k+1}})b$, where $P_{T_{k+1}}$ is the projection onto the span of the vectors $a_j, j \in T_{k+1}$.
5: $k \leftarrow k + 1$.
6: **end while**

Output: $c^* = \arg\min_{c:Supp(c) \subseteq T_k} \|b - Ac\|_2$.

Algorithm 5 Sparse Subspace Clustering by Orthogonal Matching Pursuit (SSC-OMP)(You et al. (2016b))

Input: Data $X = [x_1, \cdots, x_N], parameters k_{\max}, \epsilon$

1: Compute c_j^* from OMP(X_{-k}, x_j)
2: Set $C^* = [c_1^*, \cdots, c_N^*]$ and $W = |C^*| + |C^{*T}|$.
3: Compute segmentation from W by spectral clustering.

Output: Segmentation of data X.

Elastic net Subspace Clustering You et al. (2016a) proposed a geometric algorithm for the elastic net subspace clustering. They pointed out that the main difference of those latest clustering methods is about the choice of the regularizer $r(\cdot)$. Although elastic nets have recently been introduced for subspace clustering in Fang et al. (2014) and Panagakis and Kotropoulos (2014), the authors point out that these studies do not provide conditions that guarantee that affinity is subspace preserving, nor do they provide potential improvements in connectivity. They, therefore, give requirements for affinity-preserving subspaces and a balance between subspace preservation and connectivity properties. The objective function is as follows:

$$f(c; \mathbf{b}, A) := \lambda \|c\|_1 + \frac{1-\lambda}{2}\|c\|_2^2 + \frac{\gamma}{2}\|\mathbf{b} - Ac\|_2^2 \tag{2.21}$$

Next, normalize $\mathbf{b}$ and $\{a_j\}_{j=1}^N$ as a unit of ℓ_2 norm, and the elastic net optimize problem can be computed as follows:

$$c^*(\mathbf{b}, A) := \arg_c \min f(c; \mathbf{b}, A). \tag{2.22}$$

Then, utilize the concepts of *Oracle Point* and *Oracle Region* to present a geometric analysis:

a. *Oracle Point:*

$$\delta(\mathbf{b}, A) := \gamma \cdot (\mathbf{b} - Ac^*(\mathbf{b}, A)). \tag{2.23}$$

The dependency of the *Oracle Point* has been omitted by writing $\delta(\mathbf{b}, A)$ as δ. And the *Oracle Point* is unique because the c^* is also unique, which means the *Oracle Point* can only be calculated until the c^* has been calculated. The function c^* is defined by:

$$(1-\lambda)c^* = \tau_\lambda(A^T\delta) \tag{2.24}$$

The equation 2.24 shows that the solution c^* can be calculated straightforward when the *Oracle Point* δ is not defined.

b. *Oracle Region:*

$$\Delta(\mathbf{b}, A) := \{v \in \mathbb{R}^D : \|v\|_2 = 1, \mu(v, \delta) > \frac{\lambda}{\|\delta\|_2}\} \tag{2.25}$$

Algorithm 6 Elastic net Subspace Clustering (EnSC)(You et al. (2016a))

Input: $A = [a_1, \cdots, a_N] \in \mathbb{R}^{D\times N}, \mathbf{b} \in \mathbb{R}^D, \lambda$ and γ

1: Initialize the support set T_0 and set $k \leftarrow 0$.
2: **loop**
3: Compute $c^*(\mathbf{b}, A_{t_k})$ as the equation 2.22 by using any solver.
4: Compute $\delta(b, A_{T_k})$ from $c^*(b, A_{T_k})$ as the equation 2.24 shows.
5: Active set update: $T_{k+1} \leftarrow \{j : a_j \in \Delta(b, A_{T_k})\}$.
6: If $T_{k+1} \subseteq T_k$, terminated;otherwise set $k \leftarrow k + 1$.
7: **end loop**

Output: A vector c equals to $c_{T_k} = c^*(b, A_{T_k})$ or zeros, and its support is T_{k+1}.

Although the elastic net optimization problem has been used to solve the subspace clustering, the previous work could not be considered an efficient algorithm for the lagre scale dataset and it requires to calculate the whole data matrix A. However, the Elastic net Subspace Clustering method is proposed to reduce the scale of the problem by using the concepts of the *Oracle Region* and *Oracle Point* as the Algorithm 6 shows.

You et al.exploit the mixture of $L1$ and $L2$ norms to balance the properties of the subspace preserving and connectedness. And their analysis is built upon the optimization problem $\min_c f(c; x_j, X^t_{-j})$ and treat all points from other subspaces as newly added columns. Finally, they got the result given $x_j \in S_t$ that the vector $c^*(x_j), X_{-j}$ is subspacee preserving if and only if $x_k \notin \Delta(x_j, X^t_{-j})$ for all $x_k \notin S_t$. To ensure that the subspace is maintained, we need a small oracle region. And to ensure that connectivity is maintained, we need a large oracle region. The elastic net balances the L_1 regularization and L_2 regularization. And the oracle region will decrease as λ is increased from zero. Finally, the Elastic net Subspace Clustering algorithm finds a effective condition for the affinity for the balck between the subspace preserving and the connectivity. Let $x_j \in \mathcal{S}_l, \delta_j = \delta(x_j, x^l_{-j})$ be the *Oracle Point*. Then, the $c^*(x_j, X_{-j})$ is subspace preserving if

$$\max_{k:x_k\notin S_t} \mu(x_k, \delta_j) \leq \frac{r_j^2}{r_j + \frac{1-\lambda}{\lambda}} \tag{2.26}$$

Low-Rank Representation Liu et al. (2013) proposed a novel method named Low-Rank Representation(LRR) by seeking the lowest-rank representation and proved that the convex program in this algorithm could address the subspace clustering. To divide samples into their corresponding subspaces and simultaneously correct any faults, the authors proposed the Low-Rank Representation (LRR) method in this study to find subspace structures from corrupted data. The recently developed RPCA method, which can expand

corrupted data recovery from a single subspace to several subspaces, is generalized in LRR. The shape interaction matrix (SIM) methodology is further generalized by LRR, which offers a method for defining a SIM between two different matrices and regaining the true row space.

Let $X_0 \in \mathbb{R}^{d \times n}$ embedded in a set of n d-dimensional data points, and given a set of observation vectors is defined by:

$$X = X_0 + E_0 \tag{2.27}$$

To recover the low-rank representation matrix X_0 from the X, which is corrupted by the above equation. So, we consider to utilize the regularization for the following rank minimized problem:

$$\min_{D,E} rank(D) + \lambda \|E\|_\ell, \quad s.t. \quad X = AZ + E \tag{2.28}$$

where $\lambda > 0$ is a parameter, A is a matrix which spans the data space and $\|\cdot\|_\ell$ represents a regularization strategy. Z is the lowest-rank representation of data X_0. And the above equation is more suitable for the data points in a single low-rank subspace. However, the complexity within data collections is far greater. To handle this problem, the above rank minimization problem can be defined as:

$$\min_{Z,E} rank(Z) + \lambda \|E\|_\ell, \quad s.t. \quad X = AZ + E \tag{2.29}$$

, where the Z^* is called the minimizer also the lowest rank representation of those data points according to the dictionary A. To solve the optimized problem 2.29 effectively, it can convert to the following rank minimized problem;'

$$\min_{Z} rank(Z), \quad s.t. \quad X = AZ \tag{2.30}$$

The above solution is not unique, so the nuclear norm can replace with the rank function. The convex optimization problem is defined as:

$$\min_{Z} \|Z\|_*, \quad s.t. \quad X = AZ \tag{2.31}$$

It is shown that the nuclear norm is appropriate to replace the rank function, so a low-rank recovery to X_0 can be defined by:

$$\min_{Z,E} \|Z\|_* + \lambda \|E\|_{2,1}, \quad s.t. \quad X = AZ + E \tag{2.32}$$

, where $\ell_{2,1}$ norm represents the error term E. To optimize the below problem, it can be

convert to the equivalent problem:

$$\min_{Z,E,J} \|J\|_* + \lambda\|E\|_{2,1}, \quad s.t. \quad X = AZ + E, Z = J \tag{2.33}$$

This problem can be minimized by ALM method:

$$\mathcal{L} = \|J\|_* + \lambda\|E\|_{2,1} + tr(Y_1^T(X - AZ - E)) + tr(Y_2^T(Z - J)) + \frac{\mu}{2}(\|X - AZ - E\|_F^2 + \|Z - J\|_F^2) \tag{2.34}$$

The Algorithm 7 shows the whole process of outputting segments by using low-rank representation.

Algorithm 7 Low-Rank Representation (LRR)(Liu et al. (2013))

Input: Data matrix X, number k of subspaces.

1: Obtain the minimizer Z^* to the problem 2.32.

2: Compute the skinny SVD $Z^* = U^* \sum^* (V^*)^T$.

3: Construct an affinity matrix W by

$$[W]_{i,j} = ([\tilde{U}\tilde{U}^T]_{ij})^2. \tag{2.35}$$

4: Use W to perform NCut and segment the data samples into k clusters.

Output: Segmentation of data X.

KMeans Method The KMeans method is designed to partition data into K classes using squared Euclidean distance between the vector for any instance. Given a data set $X = \{x_1, \cdots, x_N\}, x_n \in R^d$ The data point, cluster centroid C_k in P-dimensional space is found by averaging the squared Euclidean distance value between one object over another object in a cluster. The jth variable's centroid value in cluster C_k is definde as follows:

$$\hat{x}_j^{(k)} = \frac{1}{n_k} \sum_{i \in C_k} x_{ij} \tag{2.36}$$

The centroid vector of the cluster C_k is defined by:

$$\hat{x}^{(k)} = (\hat{x}_1^{(k)}, \hat{x}_2^{(k)}, \cdots, \hat{x}_p^{(k)}) \tag{2.37}$$

Algorithm 8 shows the detailed steps of the KMeans algorithm:

Algorithm 8 KMeans Method(Singh and Reddy (2014))

Input: Data points $\boldsymbol{D}$ and number of clusters $\boldsymbol{K}$.

1: Initiate k centroids randomly.
2: Associate each data point in $\boldsymbol{D}$ with the nearest centroid and divide the data points into k clusters.
3: Recalculate the position of centroids.
4: Repeat Step 2 and Step 3 until there are no more changes in the membership of the data points in the cluster.

Output: Data points in the cluster memberships.

Chapter 3

Experiment on Synthetic Data

In this section, we measured the performance of several representative subspace clustering methods. We chose Sparse Subspace Clustering (SSC) (Elhamifar and Vidal (2012)), Sparse subspace clusterign by orthogonal matching pursuit (SSC-OMP) (You et al. (2016b)), Low Rank Representation (LRR) (Liu et al. (2013)), Elastic net Subspace Clustering (EnSC) (You et al. (2016a)), Least Squares Subspace Clustering (LSSC) (Lu et al. (2012)) and KMeans as a baseline to make the experiments. To measure the scalability of the algorithms, we measured the runtime, the clustering accuracy, NMI metrics, ARI metrics and varied the number of instances or dimensions for all algorithms. We also investigated the ability of each algorithm in determining the correct subspace for different situation by discussing the metrics of all algorithms. Finally, we picked up top 2 performing clustering methods for the next experiment. Here is the details about the metrics:

(1) Time: The time that each clustering methods spent on the dataset.

(2) Normalized Mutual Information(NMI): NMI is defined by:

$$NMI(X,Y) = \frac{I(X;Y)}{\frac{(\log k + \log c)}{2}} \tag{3.1}$$

, where X is a random variable for the cluster, Y is a random variable for the previous existing labels on the same dataset, k is the number of the clusters, and c is the number of previous existing classes. In general, an NMI value leas between 0 and 1, and the higher the NMI value is, the better the clustering quality is (Hu et al. (2009)).

(3) Adjusted Rand Index(ARI): ARI is defined by:

$$ARI(P^*, P) = \frac{\sum_{i,j}\binom{N_{ij}}{2} - \frac{[\sum_i \binom{N_i}{2}\sum_j \binom{N_j}{2}]}{\binom{N}{2}}}{\frac{1}{2}[\sum_i \binom{N_i}{2} + \sum_j \binom{N_j}{2}] - \frac{[\sum_i \binom{N_i}{2}\sum_j \binom{N_j}{2}]}{\binom{N}{2}}} \tag{3.2}$$

,where N is the number of the data points in the given dataset and $N_{i,j}$ is the number of the data points of the class label $C_j^* \in P^*$ assigned to cluster C_i in partition P. In general, an ARI value leas between 0 and 1. Only if a partition is nearly equal to the intrinsic structure and close to 0 for a random partition (Yang (2017)).

(4) Clustering accuracy: The accuracy is defined by:

$$accuracy(y, \hat{y}) = \max_{perm \in P} \frac{1}{n} \sum_{i=0}^{n-1} 1(perm(\hat{y}_i) = y_i) \tag{3.3}$$

where K is the number of the clusters and P is the set of all permutation in $[1; K]$.

3.1 Experiment on Synthetic Data

In this section, we conducted experiments on synthetic data to evaluate different subspace clustering methods: SSC, SSC-OMP, LRR, EnSC, LSSC, and KMeans as a baseline. We used time, Normalized Mutual Information(NMI) and Adjusted Rand Index(ARI) as the metrics to evaluate the performance of the algorithms. And we will compare the results of the above methods in terms of different angles between spaces, noise levels, and subspace dimensions.

3.1.1 Synthetic Data with Varying Angles

Using the normal distribution and $L1$ regularization with noise up to 0.01 in this series of experiments, we generated data from two one-dimensional subspaces embedded in a three-dimensional space. There are 200 data points in each cluster. We also adjusted the angle θ between the two subspaces during the experiment from 10 to 60 degrees,

evaluating each subspace algorithm in each case. Figure 3.1 & Figure 3.2 show the 2 subspaces rendered at 30 and 60 degrees respectively.

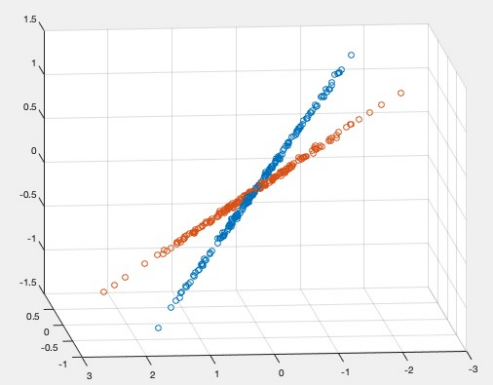

Figure 3.1: Subspace angle presented at 30 degrees

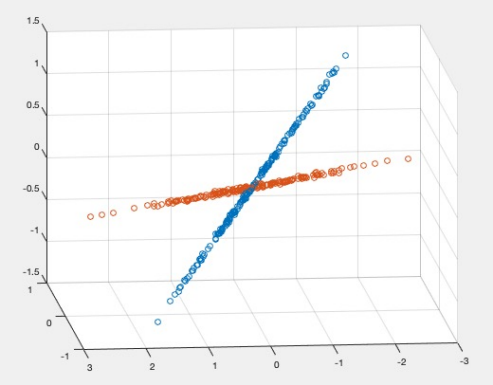

Figure 3.2: Subspace angle presented at 60 degrees

3.1.2 Synthetic Data with Varying Noise Level

Next, we exploreed the clustering performance of different clustering algorithms for different levels of noise. First, we generated 200 data points from four four-dimensional subspaces embedded in a 20-dimensional space. Different noise levels σ were then added to the data points from 0.0 to 0.5. Compared to the previous experiments, we kept the settings otherwise unchanged and increased the value in the matrix as the various noise levels.

3.1.3 Synthetic Data with Varying Dimensions

with the different subspace dimensions P_k. we set the ambient space dimension to be 20 and the subspace dimension from 2 to 16, with the noise being 0.0. The data points are generated from 4 subspaces with the number of 200.

3.2 Results

In this section, we displayed the results of six clustering algorithm on generated synthetic data and finally chose 2 clustering algorithm with top 2 clustering performance for the next step.

3.2.1 Results on Varying Angles

It can be seen in the Figure 3.3 shows that the processing time of the EnSC algorithm increases with the increase of the subspace display angle. The NMI and ARI indexes of the EnSC method increase rapidly when the subspace angle is 20 degrees, and the two indexes can reach 0.6 and 0.7 at the highest from 30 degrees. At first, the time of the KMeans method decreases with the increase of subspace angle; When the subspace angle is 40 degrees, the use time of the KMeans method gradually increases. However, with the increase of subspace angle, NMI and ARI indexes of the Kmeans method will decrease and eventually tend to zero. The overall time spent by the LRR algorithm decreases with the increase of subspace angle, showing a "sawtooth" shape. The NMI and ARI indexes of the LRR algorithm increase with the rise of subspace angle. The above situation shows that the LRR algorithm is suitable for data sets with large subspace angles. The time spent by the pedigree clustering method increases with subspace angle, and the two indexes can reach 0.95 and 1.0, respectively. The time spent by the SSC method presents an irregular "mountain peak" shape with the increase of the angle of data set zygote space, reaching the peak when the angle is 30 degrees and reaching the bottom when the angle is 50 degrees. The NMI and ARI indexes of the SSC method increase with the increase of subspace angle, and these two indexes are gradually close to 1.0. SSC method is more suitable for data sets with subspace angle greater than 30 degrees because time is less when the angle is larger, and the NMI index and ARI index are constantly approaching 1.0. The time spent by the SSC-OMP method reaches the maximum when the subspace angle is 40 degrees and then decreases continuously. When the subspace angle of the NMI and ARI index of the SSC-OMP method is equal to 40 degrees, it reaches the highest value, which is about 0.95, and then slowly decreases to about 0.87. With the increase of subspace angle, the time spent by the LSSC algorithm shows a downward trend, but it rebounds slightly. The NMI and ARI indexes of the LSSC algorithm increase with the increasing subspace angle, and these two indexes can reach 0.8 and 0.9, respectively, at the highest. It can be said that the SSC-OMP algorithm and LSSC algorithm are more suitable for data sets with slightly larger subspace angles. In this case, it takes less time and can obtain higher NMI and ARI indexes.

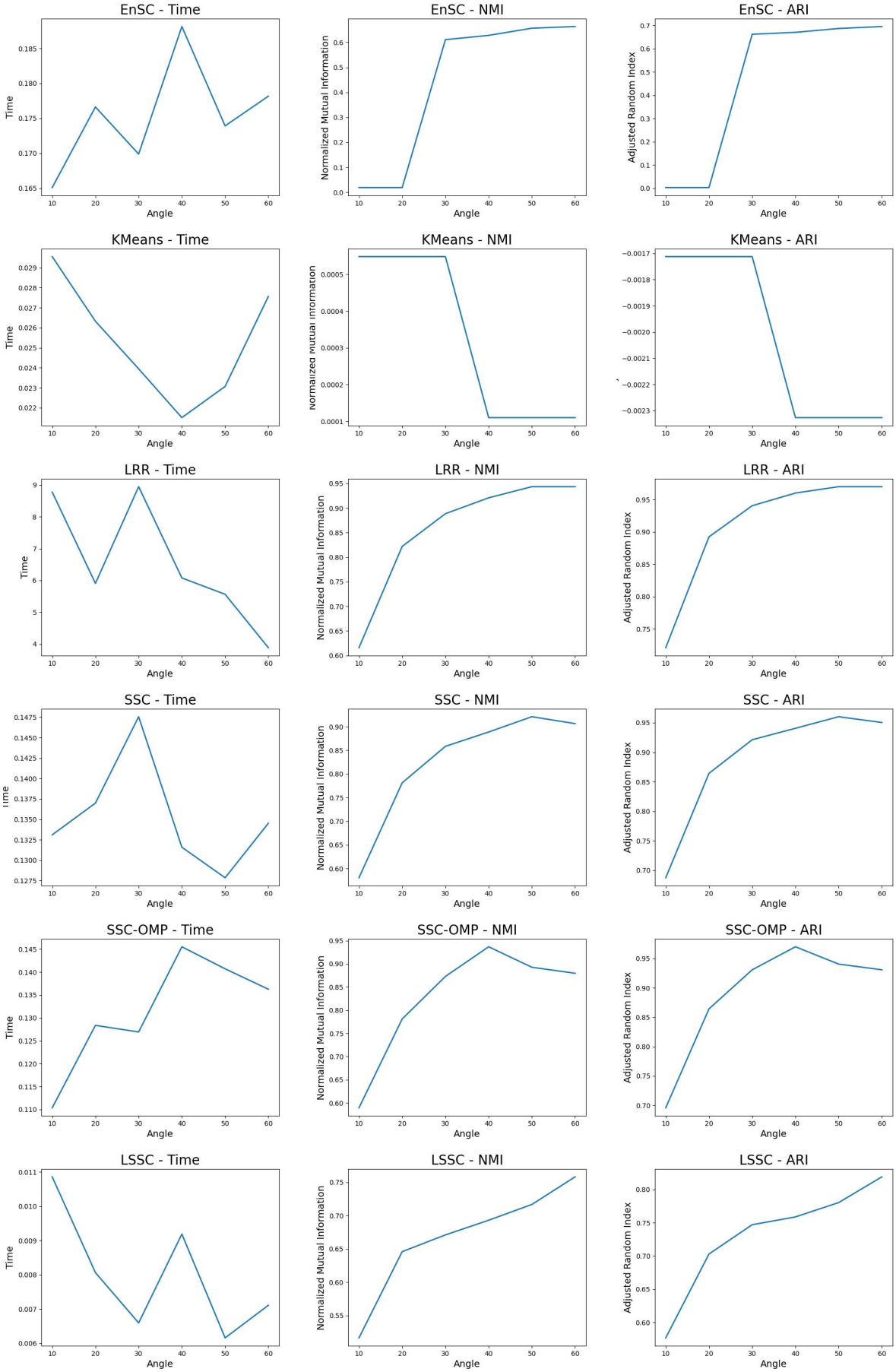

Figure 3.3: Clustering performance of six methods on varying angles

From the Table 3.1, we can obtain that only the clustering accuracy of KMeans, as a baseline, decreases as the subspace angle increases. The SSC, LRR, EnSC, LSSC and SSC-OMP algorithms all increase in clustering accuracy as the subspace angle increases. The SSC, LRR, EnSC, LSSC and SSC-OMP algorithms have the best clustering accuracy as the subspace angle increases and converge to 1.0. It also shows that the SSC, LRR and SSC-OMP algorithms are not affected by the number of the subspace angle in the data set and maintain a high level of accuracy (>0.9).

	θ=10	θ=20	θ=30	θ=40	θ=50	θ=60
KMeans	0.514	0.514	0.514	0.506	0.506	0.506
SSC	0.915	**0.965**	0.980	0.985	**0.990**	**0.987**
LRR	**0.925**	**0.972**	**0.985**	**0.990**	**0.992**	**0.992**
EnSC	0.531	0.531	0.907	0.910	0.915	0.917
LSSC	0.790	0.850	0.872	0.875	0.885	0.905
SSC-OMP	**0.917**	**0.965**	**0.982**	**0.992**	0.985	0.982

Table 3.1: Accuracy of clustering methods on varying angles

In summary, the LRR algorithm, the EnSC algorithm, the SSC algorithm, the SSC-OMP algorithm, and the LSSC algorithm increase the NMI and ARI metrics as the subspace angle in the data set continues to grow. Among them, the SSC- OMP algorithm and the LRR algorithm can eventually achieve a perfect clustering performance of nearly 1.0 for the NMI metric. For the clustering accuracy, the LRR algorithm and the SSC-OMP algorithm maintained high clustering accuracy (>0.9) across all configurations. The above analysis shows that the LRR and SSC-OMP algorithms can sustain high clustering performance in data sets with large subspace angles, which means that both algorithms are more flexible for data sets with unknown structures.

3.2.2 Results on Varying Noise Level

Figure 3.4 shows the mutual information of the time, adjusted random exponent and normalisation for each clustering method at different noise levels. For the EnSC algorithm, the time consumed reaches a peak when the noise reaches 0.1, after which the magnitude of time converges to a stable value of 1.1 seconds. The NMI and ARI metrics of the EnSC algorithm decrease as the noise level increases, eventually converging to 0, which means that EnSC is not suitable for data sets with relatively large noise levels or that it is ideal for clean data sets because the NMI metric and ARI metric of the algorithm is 1.0 at noise equals to 0. For the KMeans algorithm, the time spent decreases as the noise. The NMI

and ARI metrics of the KMeans algorithm decrease as the noise level increases. The time spent by the LRR algorithm is 'sawtooth' as the noise increases, and the overall trend decreases. The NMI and ARI metrics for the SSC algorithm decrease as the noise level increases and then level off after rapid growth from 0 to 0.1. The NMI and ARI metrics of the SSC-OMP algorithm decrease as the noise level increases, eventually converging to 0. The LSSC algorithm takes less and less time as the noise level increases. NMI and ARI metrics decrease as the noise level increases, eventually converging to 0.

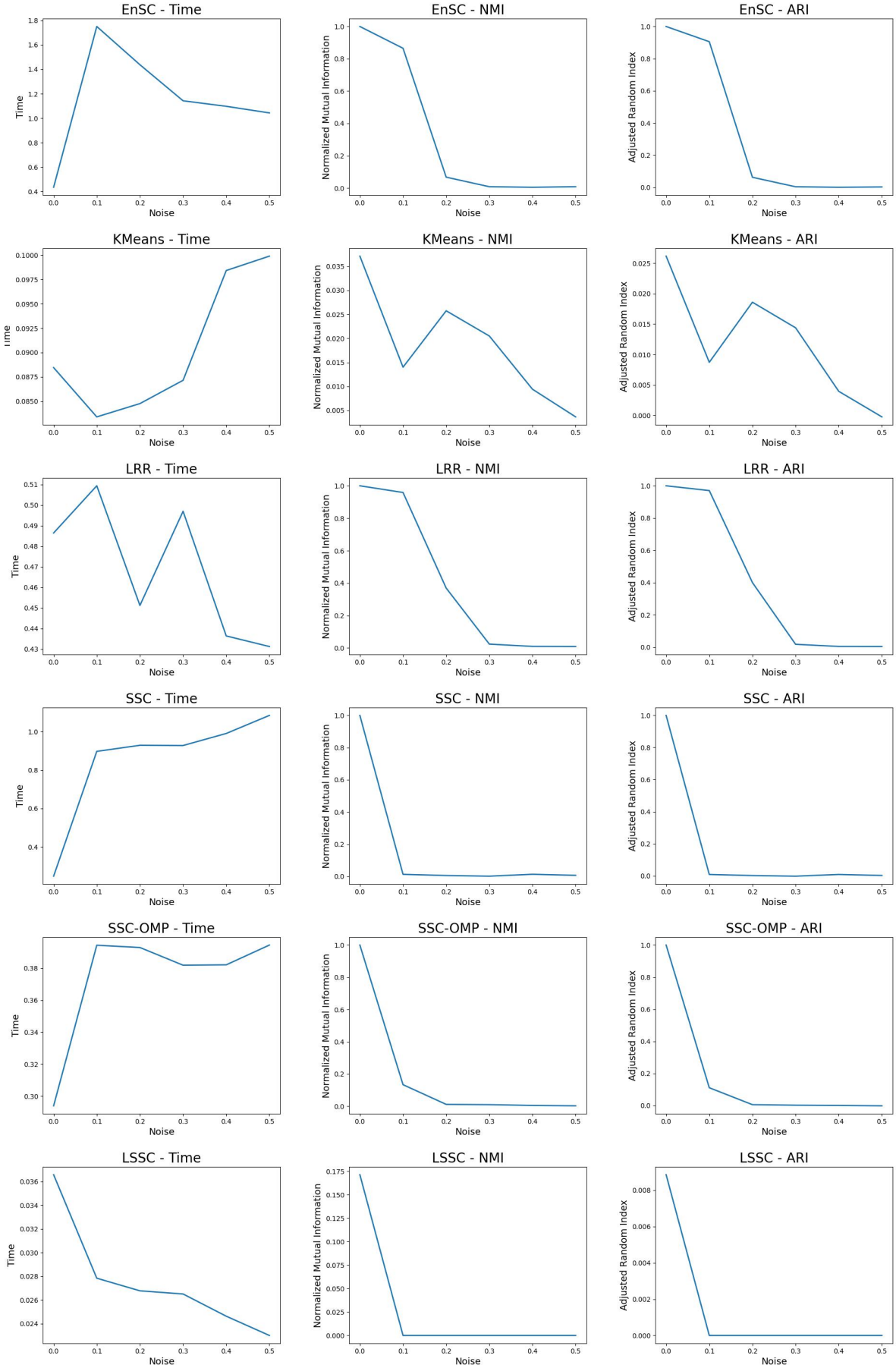

Figure 3.4: Clustering performance of six methods on varying noise levels

Table 3.2 showcases the clustering accuracy of the six clustering methods on varying noise level. The clustering accuracy of the SSC algorithm even decreases from 1.0 to 0.299 directly due to the noise going from 0 to 0.1. Each clustering algorithm seems to have a noise 'jumping off point'. For example, the noise 'jump point' for the LRR algorithm is when the noise equals 0.2, a direct reduction of almost 0.4 from the previous level, which means that the best noise level for the LRR algorithm is below 0.2. The overall performance of the LRR and EnSC algorithms in the table is better than the other four algorithms, with noise 'jump points' of 0.2 and 0.1, respectively, which may seem low. Still, overall these two algorithms outperform the other algorithms in terms of clustering accuracy for large sets of noise levels. Interestingly, the KMeans algorithm, as a baseline algorithm, outperforms even some of the subspace clustering algorithms in the case of significant noise levels, presumably because of the difference between the advantages of simple and complex algorithms in data sets with high internal confusion is not very clear.

	σ= 0.0	σ=0.1	σ=0.2	σ=0.3	σ=0.4	σ=0.5
KMeans	0.324	0.288	0.329	**0.328**	**0.291**	0.281
SSC	**1.000**	0.299	0.289	0.269	**0.306**	**0.293**
LRR	**1.000**	**0.989**	**0.724**	**0.340**	0.285	**0.293**
EnSC	**1.000**	**0.964**	**0.400**	0.294	0.285	0.283
LSSC	0.316	0.250	0.250	0.250	0.250	0.250
SSC-OMP	**1.000**	0.395	0.296	0.293	0.283	0.276

Table 3.2: Accuracy of various clustering methods on various noise levels

In summary, all six clustering algorithms performed best when the noise level was equal to 0. The clustering accuracy, NMI, and ARI metrics of the EnSC, LRR, SSC and SSC-OMP algorithms were all around 1.0 when the noise level was 0. All clustering algorithms showed varying degrees of performance degradation as the noise level increased. The NMI and ARI metrics of the LRR and EnSC algorithms do not converge to 0 until the noise level in the data set equals 0.3. The clustering accuracy of these two algorithms is consistently higher than the other algorithms as the noise level in the data set increases. Through the above analysis, data sets with high internal confusion can affect the impact of clustering algorithms to some extent. In the absence of noise, many clustering algorithms can individually perform well, but when noise is added, some clustering algorithms fail to deliver satisfactory results. The LRR and EnSC algorithms perform better than the other clustering algorithms, according to the above description.

3.2.3 Results on Varying Dimensions

Figure 3.5 shows the time, clustering accuracy, adjusted random index and normalized mutual information of each clustering method in different dimensions. When P_k is between 2 and 6, the time spent by the EnSC algorithm gradually decreases, while when P_k is between 6 and 16, the time spent by this algorithm gradually increases and decreases. When P_k is equal to 14, the time spent by this algorithm is the maximum. With the increasing of dimensions, the algorithm as a whole increase and then decreases. When P_k is equal to 4, the NMI and ARI indexes of the EnSC algorithm rise to the highest level of 1.0 and remain until P_k is equal to 12, and then the two indexes of this algorithm get into the way and decrease and approach to 0. Therefore, EnSC is suitable for data sets with P_k between 4 and 10 and can maintain a high level of clustering performance. The time spent by the KMeans algorithm is generally jagged, which means that the time spent by the KMeans algorithm seems to have nothing to do with the dimensions in the data set. When P_k equals 6, the NMI and ARI indexes of the KMeans algorithm drop sharply and finally approach 0. When P_k reaches 6, the LRR algorithm takes the most time when P_k comes to 8, and when P_k exceeds 10, it takes about 10s. When P_k is less than 10, the NMI and ARI indexes of the LRR algorithm keep the best level of 1.0 and then gradually decrease to 0 with the increase of P_k. The best applicable range of the LRR algorithm is when P_k is less than 10. The time spent by the SSC algorithm increases with the number of dimensions. The NMI and ARI indexes of SSC keep stable when P_k is less than ten and then gradually decrease to 0 with the increase of P_k. The time spent by the SSC-OMP algorithm increases with the increase of dimensionality. SSC-OMP algorithm is more suitable for data sets with P_k less than 6. At this time, this algorithm's NMI and ARI indexes remain at the level of 1.0. With the decrease of P_k, the LSSC algorithm takes less time. When P_k is greater than 4, the NMI and ARI metric of the LSSC algorithm equals 0. LSSC is not applicable when P_k is greater than 4.

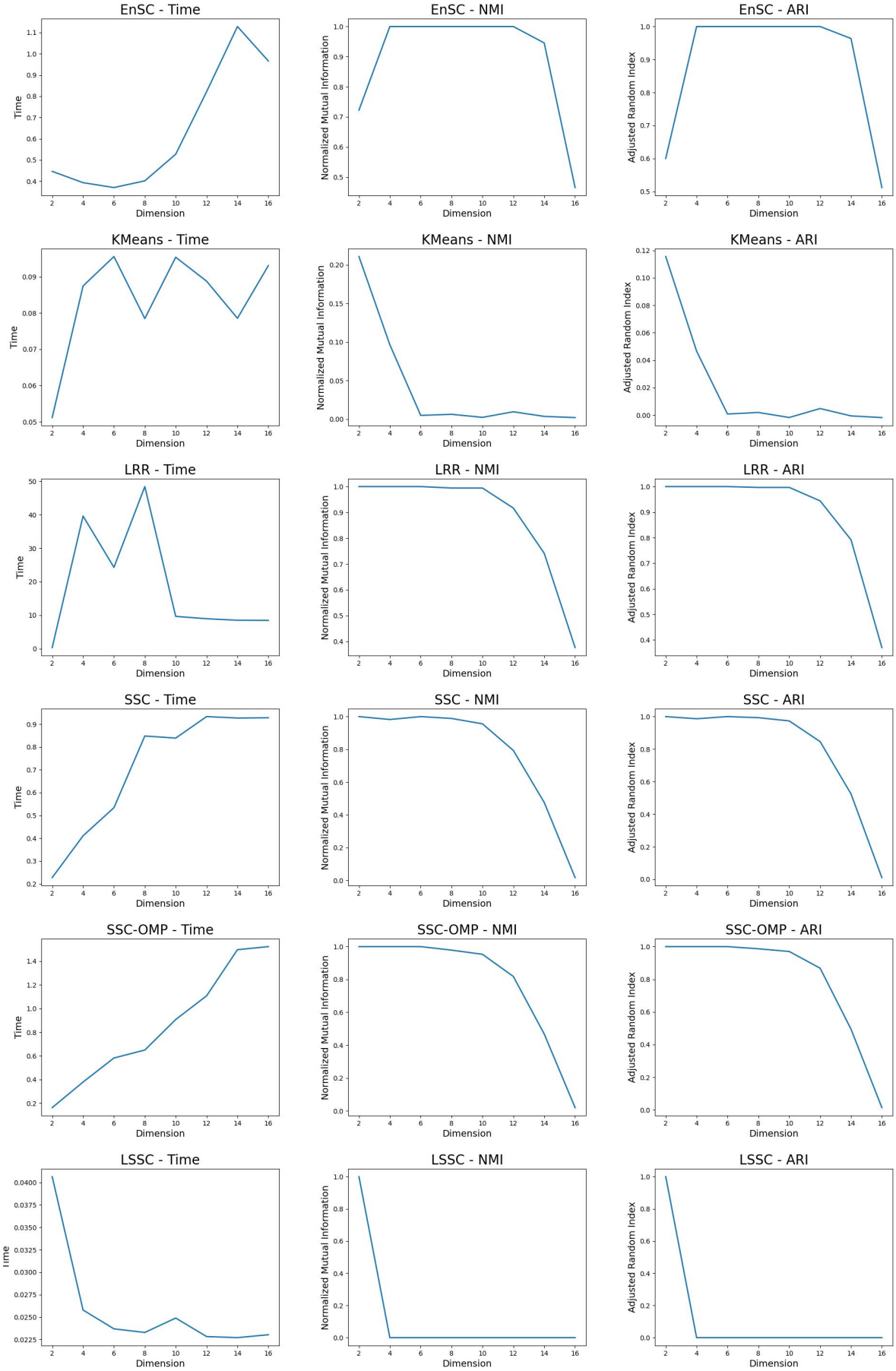

Figure 3.5: Clustering performance of six methods on varying dimensions

Table 3.3 displays the clustering accuracy of the six clustering methods on varying dimensions. All clustering algorithms have high clustering accuracy in low dimensions except the EnSC algorithm, which is able to maintain high clustering accuracy in high dimensions. The LSSC algorithm is not adapted to high dimensions, and the clustering accuracy in the case of subspace dimensions equal to 4 is directly reduced from 1 to 0.25. this indicates, to some extent, that LSSC generally applies to ultra-low dimensional data sets or clean data sets that need to be processed after dimensionality reduction.

	$P_k = 2$	$P_k = 4$	$P_k = 6$	$P_k = 8$	$P_k = 10$	$P_k = 12$	$P_k = 14$	$P_k = 16$
KMeans	0.379	0.344	0.281	0.294	0.270	0.294	0.271	0.273
SSC	**1.000**	0.995	**1.000**	0.998	0.990	0.940	**0.791**	0.308
LRR	**1.000**	**1.000**	**1.000**	**0.999**	**0.999**	**0.979**	0.688	**0.925**
EnSC	0.751	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.986**	**0.784**
LSSC	**1.000**	0.250	0.250	0.250	0.250	0.250	0.250	0.250
SSC-OMP	**1.000**	**1.000**	**1.000**	0.995	0.989	0.949	0.771	0.338

Table 3.3: Accuracy of various clustering methods on various dimensions

In summary, the KMeans and LSSC algorithms are suitable for data sets with low subspace dimensions (P_k less than 4), as they perform best when P_k equals two and the NMI and ARI metrics reach their peak. The EnSC algorithm is somewhat different in that it applies to data sets with subspace dimensions between 4 and 12, indicating that the EnSC algorithm is more suitable for data sets with higher dimensions. By looking at the clustering accuracy of each algorithm, it can be seen that the LRR algorithm and the EnSC algorithm consistently perform well as the subspace dimensions continue to change. However, it is worth mentioning that almost all clustering algorithms perform best in low-dimensional data sets. The above experiments on data sets of different dimensions also show that the ability to reduce dimensionality is an essential component of a clustering algorithm to make better use of it.

3.3 Discussion and Conclusions

According to the above analysis, it can be found that EnSC, LRR, SSC, and LSSC algorithms can maintain high NMI and ARI metrics even when the subspace angle is increased. Among them, the LRR algorithm maintains a high level of clustering accuracy above 0.9 even when the subspace angle constantly changes, indicating that the LRR algorithm requires relatively low complexity within the data set. The clustering accuracy of each clustering algorithm in the table shows that the other algorithms, except the EnSC

and LSSC algorithms, are not very sensitive to the subspace angle, and their experimental results have remained relatively stable. All six algorithms show a decrease in NMI metrics, ARI metrics and clustering accuracy as the noise level increases. However, the LRR and EnSC algorithms are more 'tolerant' of increasing noise. In this respect, it can be seen that all clustering algorithms perform better on clean data sets, which implies the importance of pre-processing the data for the data set. The "comfort zone" for the EnSC algorithm is between P_k equals 4 and 12, while the "comfort zone" for the LRR, SSC and SSC-OMP algorithms is between P_k equals 4 and 12. In terms of clustering accuracy, the EnSC and LRR algorithms consistently outperformed the other clusters as the dimensionality of the subspace changed. We can experiment with dimensionality and discover that the clustering method performs better with low-dimensional datasets. So a good dimensionality reduction operation is also an important step to help improve the clustering performance. Based on the above analysis, the **EnSC** algorithm and the **LRR** algorithm were judged to be the best among the six algorithms in terms of different subspace angles, noise levels and dimensions.

Chapter 4

A Novel Clustering Method

4.1 Method

In *Chapter 3*, we obtained that the Low Rank Representation (LRR) and Elastic net Subspace Clustering (EnSC) are the top 2 performing subspace clustering methods after experimenting with synthetic data with varying angles, noise levels and dimensions. However, the above two clustering methods use spectral clustering to complete the final clustering step. Spectral clustering helps us to overcome two major problems in clustering: one is the shape of the clusters, and the other is determining the cluster centroids. KMeans algorithms generally assume that the clusters are spherical or circular, i.e. within a k-radius of the cluster centroids, and multiple iterations can determine the cluster centroids. In spectral clustering algorithms, clusters do not follow a fixed shape or pattern. Points that are distant but connected belong to the same cluster, while points that are closer together may belong to different clusters if they are not connected, which means that the algorithm can work for data of different shapes and sizes. However, spectral clustering can be computationally expensive for large data sets, as it requires the calculation of eigenvalues and eigenvectors and then clustering. The algorithm, therefore, takes a lot of running time to run and even suffers the consequences of multiple runs with unsustainable results. Additionally, before beginning the spectral clustering process, the k-value for the number of clusters must be fixed. However, this issue can be resolved using the DBSCAN clustering algorithm. DBSCAN does not need one to predetermine the number of clusters in the data, in contrast to spectral clustering. It can even locate a cluster surrounded by other clusters but not connected to them. DBSCAN only needs two parameters and is generally unaffected by ordering points in the database because of the notion of noise in the algorithm, which is particularly robust to outliers. The DBSCAN algorithm is able to find the most suitable parameters through a certain search for parameters, in order to achieve a clustering algorithm that fits the data set and achieves the

best clustering performance. In this paper, we proposed a novel method that combines two well-performing clustering methods using the DBSCAN method to extract the final clustering instead of spectral clustering.

4.2 Algorithm

As described in Section 4.1, we have concentrated on the final clustering operation. We, therefore, propose to first apply the subspace clustering algorithm directly to the data set and obtain the solution to the convex problem and the output Affinity Matrix and then replace the traditional spectral clustering algorithm with the DBSCAN algorithm, which is more controlled to find neighbours near the centroid and does not require the input of the number of clusters.

Figure 4.1 showcases the main idea of this novel method. Through Chapter 3, we obatined the top 2 clustering performing method EnSC and LRR comparing the NMI metrics, ARI metrics and clustering accuracy of six clustering algorithm by using synthetic data. And apply this two algorithms to solve the convex optimization problems as follows:

$$\textbf{EnSC:} \quad c^*(\mathbf{b}, A) := \arg_c \min f(c; \mathbf{b}, A). \tag{4.1}$$

$$\textbf{LRR:} \quad \min_Z \|Z\|_*, \quad s.t. \quad X = AZ \tag{4.2}$$

And then DBSCAN method is replacing spectral clustering method are then applied on the affinity matrix

$$W = |C| + |C|^T \tag{4.3}$$

, where C is the solution to the convex optimization problem to obtain the final clustering.

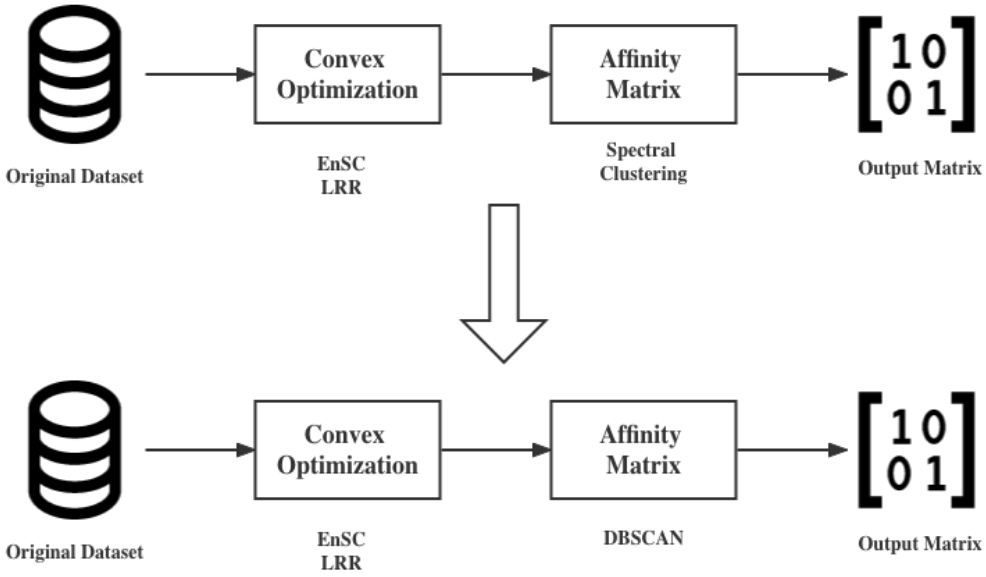

Figure 4.1: Illustration of the novel method

Spectral clustering is more costly for high-dimensional data sets because it requires knowing the exact number of clusters k in advance, calculating the eigenvalues and eigenvectors, and then performing the clustering. Spectral clustering algorithms require long run times and sometimes inconsistent results from multiple runs. As the DBSCAN algorithm is already very mature, the cost of replacing the last step of the subspace clustering algorithm is low, and the gain is considerable. In contrast to spectral clustering, DBSCAN does not require the experimenter to specify the number of clusters in the data beforehand. Since the concept of noise, boundary points, is already defined in the DBSCAN algorithm, searching for neighbouring points within a radius ε of the centroid is very efficient. In addition, DBSCAN requires only two parameters to be entered manually. The researcher can find the two parameters most optimised to achieve the best clustering algorithm for the dataset and achieve the best clustering performance. As the Figure 4.1 shows that the new method maintains the integrity of the first half of the algorithm while changing the final segmentation step for more flexibility. It is possible to choose a method with higher clustering effectiveness to solve the convex problem.

Algorithm 9 Improved EnSC method

Input: $A = [a_1, \cdots, a_N] \in \mathbb{R}^{D\times N}, \mathbf{b} \in \mathbb{R}^D, \lambda$ and γ, radius ε, density threshold *minPts*, distance function *dist*.

Initialize the support set T_0 and set $k \leftarrow 0$.
loop
  Compute $c^*(\mathbf{b}, A_{t_k})$ as the equation 2.22 by using any solver.
  Compute $\delta(b, A_{T_k})$ from $c^*(b, A_{T_k})$ as the equation 2.24 shows.
  Active set update: $T_{k+1} \leftarrow \{j : a_j \in \Delta(b, A_{T_k})\}$.
  If $T_{k+1} \subseteq T_k$, terminated;otherwise set $k \leftarrow k + 1$.
end loop
(Obtain the Affinity Matrix)
foreach point p **in** Affinity Matrix **do**
  if label(p) ≠ undefined **then continue**
  Neighbors $N \leftarrow$ RangeQuery(Affinity Matrix, dist, p, ε)
  if $|N| <$ *minPts* **then**
    label(p) ← Noise
    continue
  $c \leftarrow$ next cluster label
  label(p) $\leftarrow c$
  Seed set $S \leftarrow N\backslash\{p\}$
  foreach q **in** S **do**
    if label(q) = Noise **then** label(q) $\leftarrow c$
    if label(q) ≠ undefined **then continue**
    Neighbors $N \leftarrow$ RangeQuery(Affinity Matrix, dist, q, ε)
    label(q) $\leftarrow c$
    if $|N| <$ *minPts* **then continue**
    $S \leftarrow S \cup N$

Output: Labelled clusters

As the Algorithm 9 shows the complete EnSC algorithm using the new method. You can see that the first half uses the efficient EnSC to solve the convex problem, then the Affinity Matrix obtained as a data set is used as input to DBSCAN, and finally, the labelled data points are obtained. From this, we can also extend the definition of the algorithm with more generality, as shown in Algorithm 10.

Algorithm 10 The general algorithm

Input: Data matrix X, radius ε, density threshold $minPts$.

1: Initial the variables and set up the definition of the convex problem $f(X)$.
2: Obtain the solution of the convex problem and Affinity Matrix D.
3: Utilize DBSCAN algorithm with radius ε, density threshold $minPts$ and Affinity Matrix D.

Output: Return relabelled matrix.

The Algorithm 10 demonstrates that it is possible to select the algorithm with the best results to solve the convex problem defined by the algorithm and then obtain the Affinity Matrix for the final clustering operation with DBSCAN. The DBSCAN was chosen because it is used to separate high-density clusters from low-density clusters and is good at finding regions in the data with high-density observations. It is good at distinguishing between high and low-density clusters in a given dataset. And it is also well suited for dealing with outliers in a dataset. In this paper, we have chosen the EnSC algorithm and the LRR algorithm, and *Chapter 5* will show concrete clustering results applied to a real-world data set and compare them with the original algorithm.

Chapter 5

Experiment on Real Data

In this section, we evaluated the performance of the novel clustering method proposed from *Chapter 4* using two real-world datasets. We selected the Hopkins 155 dataset and the Extend Yale B face dataset to test the performance of the novel clustering method on a real-world dataset. In this series of experiments, we continued to use time, NMI metrics, ARI metrics and clustering performance as criteria for evaluating the performance of the new clustering method in terms of affinity matrix.

5.1 Dataset Description

This section provides description regarding the datasets that have been used in this dissertation.

5.1.1 Hopkins 155 Dataset

The Hopkins 155 dataset consists of 156 motion video sequences, each corresponding to a low-dimensional subspace and including the features derived from all frames (Tron and Vidal (2007)). This dataset contains sequences with 2 or 3 motions, which can be divided into 3 sequence categories:

1. Checkerboard: It contains 103 sequences of indoor scenes.
2. Traffic: It contains 38 sequences of outdoor traffic scenes.
3. Other(Articulated/non-rigid): It contains 13 sequences showing head, face, and people walking, etc

5.1.2 Extended Yale B

The Extended Yale B database contains 16128 images of 28 human subjects under 9 poses and 64 illumination conditions (Georghiades et al. (2001)). These images were taken under different lighting conditions and with a variety of facial expressions.

Figure 5.1: The Extended Yale B face dataset

First, we imported the Hopkins 155 and Extended Yale B data sets and separated the image data from the labelled data. The unimproved and improved EnSC and LRR algorithms were then applied to the image dataset. The improved EnSC algorithm and LRR algorithm will solve the convex problem and then apply the output affinity matrix to the spectral clustering algorithm and DBSCAN method, respectively. Finally, the corresponding labelled data from the experimental results are output and compared with the actual labelled data to produce the corresponding runtime, NMI metrics, ARI metrics and clustering accuracy to evaluate the performance of each clustering algorithm, as shown in Figure 5.1, Figure 5.3, 5.2 and Figure 5.4.

5.2 Experiment Result

5.2.1 Hopkins 155 Dataset

	Time	NMI	ARI	Accuracy
Spectral Clustering	1.732	0.580	**0.135**	0.200
DBSCAN	**1.656**	**0.683**	0.088	**0.358**

Table 5.1: New EnSC method applied on Extended Yale B

	Time	NMI	ARI	Accuracy
Spectral Clustering	0.858	0.211	0.018	0.097
DBSCAN	**0.806**	**0.633**	**0.045**	**0.372**

Table 5.2: New LRR method applied on Extended Yale B

5.2.2 Extended Yale B

	Time	NMI	ARI	Accuracy
Spectral Clustering	28.884	**0.645**	**0.238**	0.207
DBSCAN	**23.597**	0.563	0.029	**0.220**

Table 5.3: New EnSC method applied on Extended Yale B

	Time	NMI	ARI	Accuracy
Spectral Clustering	16.650	0.135	**0.025**	**0**.089
DBSCAN	**16.333**	**0.407**	0.009	**0.193**

Table 5.4: New LRR method applied on Extended Yale B

5.2.3 Discussion

As Figure 5.1 and 5.2 show that the improved EnSC algorithm with a 10% higher NMI metric and 15.8% higher clustering accuracy for the Hopkins 155 data set than before the improvement, and the improved LRR algorithm with a 42.2% higher NMI metric, 2.7% higher ARI metric and 27.5% higher clustering accuracy for the Hopkins 155 data set than before the improvement. As Figure 5.4 shows that the improved LRR algorithm has a 27.2% higher NMI metric and a 10.4% higher clustering accuracy for the Extend

Yale B data set than before the improvement. From the above analysis, it can be learned that using the DBSCAN algorithm as the last step of the clustering operation can perform better than the traditional clustering algorithm that uses the spectral clustering algorithm as the final step of the clustering operation. This is because the spectral clustering algorithm requires inputting the number of clusters in advance. In contrast, the algorithm can adjust the DBSCAN algorithm to find the most suitable parameters to achieve optimal clustering performance.

Chapter 6

Conclusions & Future Work

This chapter summaries this dissertation work from following three aspects: First, we discussed the conclusion of this dissertation. Second, we talked about the challenge faced in this thesis. Third, we proposed the future work of this project.

6.1 Conclusion

This dissertation aimes to develop a novel clustering method to improve the clustering performance of high-dimensional datasets. We further learned about the idea of modern clustering methods and implemented them in Python. And we generate the synthetic data on varying angles, noise levels, and dimensions. Then we evaluated the clustering performance of modern clustering methods using synthetic data and analysed the result deeply to obtain the top-performing clustering method in a different situation. Next, we pointed out that most subspace clustering methods use spectral clustering to deal with the affinity matrix, which needs the exact number of clusters in advance. And this thesis also discussed the advantages and disadvantages of using spectral clustering as the final clustering step and the advantage of using DBSCAN instead. We described a novel method that combines two well-performing clustering methods using the DBSCAN method to extract the final clustering instead of spectral clustering.

6.2 Challenge

During this dissertation work, many technical challenges have been faced. For example, spent a lot of time understanding complex and difficult clustering concepts and implementing modern clustering algorithms. And the parameters of the clustering algorithm are very tricky to choose and need to be chosen using scientific methods to conduct ex-

periments. There are technical challenges, such as how to deal with the lack of significant experimental results.

6.3 Future Work

For this dissertation, there is still valuable work to be done. Firstly, in the Convex Problem treatment, more modern clustering algorithms can be used for the selection, such as Block Diagonal Representation (BDR), Structured Sparse Subspace Clustering (S3C), Affine Sparse Subspace Clustering (ASSC), Weighted Sparse Subspace Representation (WSSR) algorithms, etc. In addition, for the treatment of Affinity Matrix, we have only compared two algorithms here, Spectral Clustering and DBSCAN; there are still other algorithms, such as the Boost algorithm, that can be used for comparison experiments. In addition, the parameter selection method for the DBSCAN algorithm is not scientific enough, resulting in a final experimental result that is not as significant as desired.

Bibliography

Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 94–105.

Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499. Santiago, Chile.

Anand, S., Mittal, S., Tuzel, O., and Meer, P. (2013). Semi-supervised kernel mean shift clustering. *IEEE transactions on pattern analysis and machine intelligence*, 36(6):1201–1215.

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799.

Defays, D. (1977). An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366.

Donath, W. E. and Hoffman, A. J. (2003). Lower bounds for the partitioning of graphs. In *Selected Papers Of Alan J Hoffman: With Commentary*, pages 437–442. World Scientific.

Elhamifar, E. and Vidal, R. (2012). Sparse Subspace Clustering: Algorithm, Theory, and Applications.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

Fang, Y., Wang, R., Dai, B., and Wu, X. (2014). Graph-based learning via auto-grouped sparse regularization and kernelized extension. *IEEE Transactions on Knowledge and Data Engineering*, 27(1):142–154.

Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305.

Friedman, J. H. (1994). An overview of predictive learning and function approximation. *From statistics to neural networks*, pages 1–61.

Frigui, H. and Krishnapuram, R. (1997). Clustering by competitive agglomeration. *Pattern recognition*, 30(7):1109–1119.

Fukunaga, K. and Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40.

Georghiades, A., Belhumeur, P., and Kriegman, D. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660.

Hagen, L. and Kahng, A. B. (1992). New spectral methods for ratio cut partitioning and clustering. *IEEE transactions on computer-aided design of integrated circuits and systems*, 11(9):1074–1085.

Hu, X., Zhang, X., Lu, C., Park, E. K., and Zhou, X. (2009). Exploiting wikipedia as external knowledge for document clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 389–396.

Jahirabadkar, S. and Kulkarni, P. (2013). Clustering for high dimensional data: density based subspace clustering algorithms. *International Journal of Computer Applications*, 63(20).

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323.

Jia, H., Ding, S., Xu, X., and Nie, R. (2014). The latest research progress on spectral clustering. *Neural Computing and Applications*, 24(7):1477–1486.

Kailing, K., Kriegel, H.-P., and Kröger, P. (2004). Density-connected subspace clustering for high-dimensional data. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 246–256. SIAM.

Kriegel, H.-P., Kröger, P., and Zimek, A. (2012). Subspace clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(4):351–364.

Kriegel, H.-P., Kröger, P., Sander, J., and Zimek, A. (2011). Density-based clustering. *WIREs Data Mining and Knowledge Discovery*, 1(3):231–240. _eprint: https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/widm.30.

Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y., and Ma, Y. (2013). Robust Recovery of Subspace Structures by Low-Rank Representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1):171–184. arXiv: 1010.2955.

Lu, C.-Y., Min, H., Zhao, Z.-Q., Zhu, L., Huang, D.-S., and Yan, S. (2012). Robust and efficient subspace segmentation via least squares regression. In *European conference on computer vision*, pages 347–360. Springer.

Ma, X. and Dhavala, S. (2018). Hierarchical clustering with prior knowledge. *arXiv preprint arXiv:1806.03432*.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations.

Madhulatha, T. S. (2012). An overview on clustering methods. *arXiv preprint arXiv:1205.1117*.

Murtagh, F. and Contreras, P. (2012). Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97.

Murtagh, F. and Contreras, P. (2017). Algorithms for hierarchical clustering: an overview, ii. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(6):e1219.

Panagakis, Y. and Kotropoulos, C. (2014). Elastic net subspace clustering applied to pop/rock music structure analysis. *Pattern Recognition Letters*, 38:46–53.

Parsons, L., Haque, E., and Liu, H. (2004). Subspace clustering for high dimensional data: a review. *Acm sigkdd explorations newsletter*, 6(1):90–105.

Popat, S. K. and Emmanuel, M. (2014). Review and comparative study of clustering techniques. *International journal of computer science and information technologies*, 5(1):805–812.

Rani[1], Y. and Rohil, H. (2013). A study of hierarchical clustering algorithm. *ter S & on Te SIT*, 2:113.

Rodriguez, A. and Laio, A. (2014). Clustering by fast search and find of density peaks. *science*, 344(6191):1492–1496.

Rohlf, F. (1973). Algorithm 76. hierarchical clustering using the minimum spanning tree. *Comput. J.*, 16:93–95.

Saxena, A., Pal, N. R., and Vora, M. (2010). Evolutionary methods for unsupervised feature selection using sammon's stress function. *Fuzzy Information and Engineering*, 2(3):229–247.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., and Xu, X. (2017). Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905.

Sibson, R. (1973). Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34.

Singh, D. and Reddy, C. (2014). A survey on platforms for big data analytics. *Journal of Big Data*, 2.

Sokal, R. R. (1966). Numerical taxonomy. *Scientific American*, 215(6):106–117.

Strang, G. (2006). *Linear algebra and its applications.* Belmont, CA: Thomson, Brooks/Cole.

Tron, R. and Vidal, R. (2007). A benchmark for the comparison of 3-d motion segmentation algorithms. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416.

Voorhees, E. M. (1986). Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing & Management*, 22(6):465–476.

Wilson, H., Boots, B., and Millward, A. A. (2002). A comparison of hierarchical and partitional clustering techniques for multispectral image classification. In *IEEE International Geoscience and Remote Sensing Symposium*, volume 3, pages 1624–1626. Ieee.

Yang, Y. (2017). Chapter 3 - temporal data clustering. In Yang, Y., editor, *Temporal Data Mining Via Unsupervised Ensemble Learning*, pages 19–34. Elsevier.

You, C., Li, C.-G., Robinson, D. P., and Vidal, R. (2016a). Oracle based active set algorithm for scalable elastic net subspace clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3928–3937.

You, C., Robinson, D., and Vidal, R. (2016b). Scalable sparse subspace clustering by orthogonal matching pursuit. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.