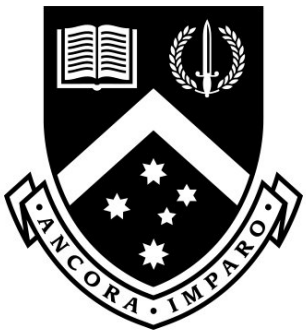
# Efficient Identification of Arbitrarily Shaped and Varied Density Clusters in High-dimensional Data




**Ye Zhu**

Supervisor: Prof. Kai Ming Ting and Dr Mark Carman

Faculty of Information Technology

Monash University

This dissertation is submitted for the degree of

*Doctor of Philosophy*

Clayton Campus May 2017

I would like to dedicate this thesis to my loving parents …

# Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Ye Zhu

May 2017

# Acknowledgements

I would like to express my deep gratitude and appreciation to my two supervisors: Prof. Kai Ming Ting and Dr Mark Carman for continuous guidance throughout the PhD journey over three years. Without them, I would never have taken a great interest in the research area of machine learning and data mining, and would never have been able to become a skilled and mature researcher. Their considerable insights and constructive comments have greatly improved the quality of this dissertation. In particular, I would like to thank Kai Ming for providing me with financial support through a grant from the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AOARD #FA2386-13-1-4043).

I gratefully thank Julie Holden for her invaluable discussion and suggestions regarding my writing of this dissertation. I would additionally like to thank professional accredited editor Mary-Jo O'Rourke AE for her help in proofreading this dissertation. Their professional writing guidance and experience have allowed me to fully express the concepts behind this research project.

In addtion, I would like to extend my sincere appreciation to all other academic and administrative staff at the Faculty of Information Technology. I thank Prof. Geoff Webb, Prof. Balasubramaniam Srinivasan, A/Prof. Chung-Hsing Yeh, Dr David Albrecht and Dr Gholamreza Haffari for being my panel members and for their constructive comments and feedback on my research during milestone seminars.

Importantly, I would like to express thanks to my loving parents, who raised me with endless love. I would not have been able to study overseas without their great support. I also would like to thank my grandma, my sister and many more friends who have always supported and encouraged me during past years. Special thanks go to my buddy, Zhi Liu, for his continual inspiration, encouragement and enduring patience.

# Publications

Publications arising from this thesis are listed as follows.

- The main material in Chapter 3 has been published in *Pattern Recognition*.

  Ye Zhu, Kai Ming Ting and Mark J. Carman. 2016. Density-ratio based clustering for discovering clusters with varying densities. *Pattern Recognition*, 60, C (December 2016), 983–997. DOI: http://dx.doi.org/10.1016/j.patcog.2016.07.007

- There are three papers utilising the theory and results from Chapter 3.

  Kai Ming Ting, Ye Zhu, Mark Carman, Yue Zhu and Zhi-Hua Zhou. 2016. Overcoming key weaknesses of distance-based neighbourhood methods using a data dependent dissimilarity measure. *In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 1205–1214. DOI: https://doi.org/10.1145/2939672.2939779

  Kai Ming Ting, Ye Zhu, Mark Carman, Yue Zhu, Zhi-Hua Zhou and Takashi Washio. 2017. Lowest probability mass neighbour algorithms: breaking loose from the metric constraint in distance-based neighbourhood algorithms. Under review by *Machine Learning*.

Bo Chen, Kai Ming Ting, Takashi Washio and Ye Zhu. 2017. Local contrast as an effective means to robust clustering against varying densities. Under review by *Machine Learning*.

- The material in Chapter 4 will be submitted to an international conference.

  Ye Zhu, Kai Ming Ting and Mark J. Carman. 2017. Grouping Points by Shared Subspaces for Effective Subspace Clustering.

# Abstract

We are living in the era of a data explosion, where clustering has become one of the most important automatic data labelling techniques used to both understand and describe the world. Clustering aims to discover the natural groupings of a set of objects which are otherwise hidden in the data. This thesis focuses on density-based clustering techniques. In contrast to other kinds of clustering methods, such as traditional partitioning and hierarchical methods, which can only discover globular-shaped clusters, density-based clustering has unique characteristics including finding clusters of arbitrary sizes and shapes while effectively separating noise and outliers. Therefore, density-based clustering has received substantial attention in both theory and practice.

Density-based algorithms suffer from one or both of the following problems. First, only a few existing algorithms can find clusters with widely varied densities and only under limited conditions. Second, they cannot effectively find clusters in high-dimensional data sets because of the "curse of dimensionality" and the presence of irrelevant attributes. The classic density-based clustering algorithm, DBSCAN, suffers from both of these problems. A number of variations of the original DBSCAN algorithm have been proposed to improve its performance on some aspects, but they still cannot solve both problems simultaneously.

This research project aims to meet this challenge by developing efficient and effective methods which enable density-based clustering algorithms to detect clusters with different densities and shapes in high-dimensional data. I have investigated and designed different approaches for each problem. The effectiveness of these proposed approaches has been verified with extensive empirical evaluations on synthetic and real-world datasets.

To overcome the first problem, this thesis identifies and analyses the condition under which density-based clustering algorithms fail to find all clusters of differing densities, and proposes three different approaches targeted to reconditioning density-based clustering algorithms. Two of these approaches are based on density-ratio estimation and the third approach is based on a data-dependent dissimilarity. All three approaches can retain the same time and space complexities as an existing density-based clustering algorithm.

To tackle the second problem, this thesis proposes an efficient and effective subspace clustering framework named *CSSub* (Clustering of Shared Subspaces). It enables similar core points to be clustered together based on the number of subspaces they share. It explicitly splits the candidate subspace selection and clustering of similar points into two separate processes, enabling different types of cluster definitions to be employed easily. It has the ability to detect clusters with varied densities which exist in different subspaces. If there are multiple clusters with varied densities detected in a subspace, the approach proposed for solving the first problem can be incorporated.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

*The observation of and the search for similarities and differences are the basis of all human knowledge.*

ALFRED B. NOBEL

We are living in the era of a data explosion: there there are tens of thousands of pieces of data generated every second. The data generated includes digital images, emails, videos and tweets [76]. Due to the increase in data size and types of data, traditional manual data labelling by humans has become an impossible task. Clustering, as the most common unsupervised knowledge discovery technique, has become one of the most popular automatic data-labelling techniques. It has been widely studied for data mining and knowledge discovery [39]. The goal of clustering is to partition a set of data points into a set of homogeneous groups based on their similarity [29]. From the perspective of pattern recognition, clusters are hidden patterns to be discovered and represent a meaningful data concept [29].

Clustering plays a crucial role in how humans understand and describe the world. For example, clustering can be used to analyse a cancerous dataset such that different cells types from a biological tissue sample can be identified and separated for further investigation. It also can be used for market research by breaking down market trends and customer habits into meaningful segments according to their shared characteristics. Clustering has become a fundamental data-analysis technique and has been applied

in a wide variety of areas such as engineering, computer science, social sciences and economics [25].

The objective of a clustering algorithm is to group data points from an unlabelled dataset into a finite and discrete set according to a "natural" hidden data structure with little or no prior information [15]. Since MacQueen [53] first proposed the *k*-Means clustering algorithm, different kinds of clustering algorithms have been studied in data mining and machine-learning communities over the past few decades, e.g., partition-based, density-based and hierarchy-based [29].

There is no universal clustering method that can effectively tackle all sorts of cluster structures, because cluster analysis generally is a difficult problem due to the subjectivity of clustering process, i.e., a dataset can be clustered in various ways based on different purposes or views [70]. All algorithms contain bias. A clustering algorithm is developed for a particular task with specific assumptions in favour of the application of interest. Thus, a clustering algorithm will inevitably perform poorly in other tasks which do not meet the corresponding premise [77].

To obtain a reliable clustering result, cluster analysis often needs effective supervision. It can use a series of trials and repetitions to support supervision in various circumstances. Domain knowledge and visual analysis can also improve the insights about the quality of the clustering. However, it is not possible to create an automated computer program which has the same intuitive insights as humans because of the significant complexity in developing the program. Moreover, outliers, irrelevant attributes and missing attribute values can make the clustering problem more complex, especially in the interpretation of the clusters. Since there is no uniform criterion to guide the selection of features and clustering schemes, we should select clustering algorithms based on the task and make an assumption of some structures among the data points.

## 1.1 Motivation

There are different kinds of clustering algorithms depending on the specific assumption and model used. Density-based clustering algorithms define clusters as regions of high density which are separated by regions of low density; the clusters are identified by grouping points above a global density threshold. In contrast to traditional partitioning methods, which can only discover globular clusters, density-based clustering has unique characteristics including finding clusters of arbitrary sizes and shapes while effectively separating noise and outliers. Therefore, density-based clustering has received substantial attention in theory and practice.

Most existing density-based clustering algorithms suffer at least one of two types of problems. First, only a few of them can find clusters with widely differing densities and only under limited conditions. Second, a high-dimensional dataset[1] normally makes it difficult to identify a proper density threshold due to the "curse of dimensionality" [24] and irrelevant attributes. Therefore, most density-based clustering algorithms are limited to low-dimensional datasets.

The classic density-based clustering algorithms, such as DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [22] and DENCLUE (DENsity-based CLUstEring) [31], model the data distribution of a given dataset using a density estimator and then apply a threshold to identify "core" points which have densities higher than the threshold. A linking method is employed to link all neighbouring core points to form a cluster.

It is well-known that DBSCAN suffers from both problems mentioned above. Although a number of variations of the original DBSCAN algorithm have been proposed to improve its performance on some aspects, they still cannot solve both problems simultaneously. Hence, it is important to develop some efficient and effective methods which enable density-based clustering algorithms to detect clusters with different

[1]There is no consistent definition of "high-dimensional datasets" [7]. Traditionally for clustering, a high-dimensional dataset is described by 10 or more attributes, because most clustering algorithms based on a traditional distance measure cannot effectively deal with these [29]. Following the same tradition, I refer to a dataset as being "high dimensional" when it has 10 attributes or more.

densities and shapes in high-dimensional data. This research project aims to meet this challenge.

## 1.2 Research Questions and Contributions

In order to mine high-dimensional datasets and to discover clusters of arbitrary shape and density, this PhD project aims to overcome the two problems of density-based clustering mentioned above.

The research questions for this PhD research are given as follows:

**Research Question 1:** Can density-based clustering algorithms be made robust for datasets containing clusters of varied densities?

**Research Question 2:** Can a density-based subspace clustering algorithm be designed which is scalable to high-dimensional datasets?

In the process of answering the two research questions, I have first analysed the underlying cause of the two problems suffered by density-based clustering algorithms, and then investigated and designed possible approaches for each problem. I have verified the effectiveness of these proposed approaches with extensive empirical evaluations on synthetic and real-world datasets.

Overall, this thesis makes the following contributions:

(1) It identifies the condition under which most density-based clustering algorithms fail to find all clusters of differing densities. (Research Question 1)

(2) It provides three approaches to overcoming the weakness of detecting clusters with varied densities. Two of these approaches (ReCon and ReScale) are based on density-ratio estimation, and the third approach (ReMass) is based on a data-dependent dissimilarity. (Research Question 1)

(3) It develops a clustering framework based on shared subspaces (*CSSub*) which allows for effective and efficient subspace clustering for high-dimensional datasets. (Research Question 2)

(4) It proposes two subspace clustering algorithms, $CSSub_D$ and $CSSub_I$, based on two different subspace scoring functions with the *CSSub* framework. (Research Question 2)

It is worth noting that, incorporated with the approach proposed for solving the first problem, *CSSub* can simultaneously identify the clusters of arbitrary shape and varied density in high-dimensional data.

## 1.3 Structure of the Thesis

This thesis is structured as follows: Chapter 2 reviews the relevant literature on different clustering algorithms and highlights their limitations which motivate this project. In this chapter, I provide the basic concepts of different kinds of clustering algorithms and categorise them into two classes, i.e., traditional and subspace clustering algorithms. Then I summarise their limitations vis-a-vis detecting clusters with arbitrary densities and shapes, before elaborating on the motivation of this thesis. I also discuss the clustering validation method chosen for this project. Chapter 3 focuses on tackling the problem of varied densities across clusters and provides three approaches. It first identifies and analyses the condition under which most density-based clustering algorithms fail to find all clusters of differing densities. After that, I propose three approaches (ReCon, ReScale and ReMass) targeted to reconditioning density-based clustering algorithms to solve this problem. Chapter 4 presents a novel clustering framework based on shared subspaces (*CSSub*) for clustering high-dimensional data. I also design two subspace clustering algorithms, $CSSub_D$ and $CSSub_I$, within this framework. I verify the effectiveness of these proposed approaches with extensive empirical evaluations on synthetic and real-world datasets in the last sections of both Chapter 3 and Chapter 4. Conclusions and some directions for further work are provided in Chapter 5.

# Chapter 2

# Literature Review

*Essentially, all models are wrong, but some are useful.*

GEORGE E. P. BOX

There are many applications of clustering algorithms designed to solve the widely diverse problems in different industries. Typically, these algorithms can be classified into several categories based on their characteristics [1]. Technically, there are probabilistic techniques, density-based techniques, distance-based techniques, spectral-based techniques and dimensionality-reduction based techniques. In terms of the data being clustered, there are various kinds of data types with different properties, such as time series data, categorical data and numerical data. Each data type requires a specific methodology for the clustering process.

In this chapter, I review classic and state-of-the-art clustering algorithms, categorised into two main classes, i.e., traditional and subspace clustering algorithms, based on their abilities in handling high-dimensional datasets. Since density-based algorithms can usually be applied on numerical datasets only, this review focuses on different kinds of algorithms used for clustering numerical datasets.

After presenting the techniques used in different kinds of algorithms, I discuss their limitations and challenges in terms of detecting clusters with arbitrary densities and shapes, before elaborating on the motivation of this thesis.

In addition, I present some popular clustering validation measures for evaluating the quality of clustering results, categorised into two main types, i.e., external clustering

validation and internal clustering validation. Finally, I discuss their usability and feasibility for empirical evaluation of my proposed clustering algorithms.

## 2.1 Traditional Clustering Algorithms

Traditional clustering algorithms are clustering algorithms in which the similarity calculation is based on full-space or all-feature information (i.e., the entire set of features describing each data point are used to determine the distance between data points). This kind of clustering algorithm can be generally categorised as partitioning methods, hierarchical methods, probabilistic-model based methods and density-based methods.

### 2.1.1 Partitioning Methods

Partitioning clustering methods are the simplest and most fundamental clustering methods. They are relatively fast, and easy to understand and implement. They organise the data points into $k$ non-overlapping partitions where each partition represents a cluster and each point only belongs to one cluster [29].

There are different criteria for evaluating the quality of a partition result. The traditional criterion required for a good clustering result is that the points from each cluster are close to each other while points from different clusters lie far apart from one another.

In order to avoid exhaustive enumeration of all possible partitionings and to discover the global optimal one, most algorithms adopt heuristic methods greedily searching for a local optimum. The most popularly used partitioning algorithms are $k$-means and $k$-medoids. They iteratively improve their partitioning results and perform well for detecting globular clusters.

#### $k$-means

The $k$-means algorithm was the first algorithm which attempted to find a feasible method for optimal partitioning [53]. The output of this algorithm is to define $k$

centroids, each of which represents a cluster. The algorithm is mainly composed of the following four steps:

1. Randomly choose $k$ points from the dataset. These points represent the initial group centroids.
2. Form $k$ clusters by assigning each point to its closest centroid.
3. When all points have been assigned, recalculate the positions of the $k$ centroids, which are the mean values of the points for each cluster.
4. Repeat Steps 2 and 3 until the centroids no longer change.

Given a dataset $D$ with $n$ points $x \in R^d$, the partitioning method will group all points into $k$ clusters $C_1, C_2, ...C_k$ such that $C_i \subset D$ and $C_i \cap C_j = \emptyset$. Let $c_i \in R^d$ be the centroid (mean) of a cluster; the objective function can be simply defined as a sum of squared error (SSE) function as below:

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} dist(x, c_i)^2 \tag{2.1}$$

where $dist(x, c_i)$ is a distance measure between a data point $x$ and centroid $c_i$.

The $k$-means clustering algorithm is basically an optimisation algorithm with the aim of minimising the objective function, resulting in clusters as compact and separated as possible.

Although the procedure will always terminate with the time complexity of $O(kin)$ (where $n$ is the number of data points, $i$ is the number of iterations, and $k$ is the number of clusters), it usually cannot find the globally optimal configuration. The $k$-means is particularly sensitive to the initial cluster centres, which are usually randomly chosen from the given dataset. It can also perform poorly on datasets containing noise and outliers which are far from the majority of the data and thus substantially affect the mean value calculation.

#### *k*-medoids

There are a few of variations of the *k*-means which choose different representative prototypes for the clusters. The *k*-medoids is a variation which is more resilient to noise and outliers than the *k*-means.

Instead of using the mean value to represent a cluster, the *k*-medoids algorithm [38] chooses one data point $c_i \in C_i$ as the centre (medoid) for the cluster. A medoid is the point with the minimum average distance to all the points in the cluster. Then the objective function is to minimise the sum of the dissimilarities between each point in the cluster and its corresponding medoid.

Since the *k*-medoids has a computational complexity of $O(ki(n-k)^2)$, which is much higher than that of the *k*-means, it is not suitable for clustering large datasets. In order to overcome this limitation, a sampling method combined with *k*-medoids, called the Clustering LARge Application (CLARA) algorithm, was proposed by Hopke and Kaufman (1990). It applies the *k*-medoids on many samples and returns the set of optimal medoids.

### 2.1.2 Hierarchical Methods

Partitioning clustering usually requires a user to define the number of clusters. Model-selection techniques can be used to overcome this challenge for *k*-means based algorithms. Alternatively, to overcome this issue hierarchical methods partition data into groups of different levels by developing a binary tree-based data structure, named a dendrogram, where different levels of the tree represent each different granularity of the clustering solution [29]. Hierarchical methods can be categorised into agglomerative and divisive methods, depending on how the hierarchical representation is created. However, the problem with this kind of clustering is that choosing a proper dissimilarity measure and choosing a proper level for cluster analysis are both difficult.

Figure 2.1 shows the application of an agglomerative hierarchical clustering method and a divisive hierarchical clustering method on a dataset of five points.

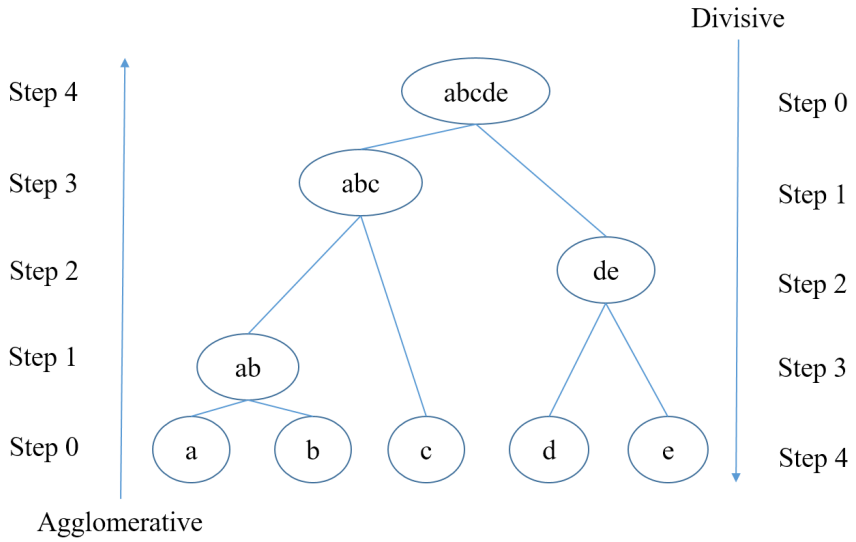



Fig. 2.1 Agglomerative and divisive hierarchical clustering on a dataset with 5 points.

### Agglomerative Methods

Agglomerative methods build a hierarchical representation bottom up such that individual points are merged successively to form a tree-based structure based on a dissimilarity measure. Two clusters are merged during each iteration and all points become a single cluster at the end. There are different distance measures that can be used to merge two clusters by trading off quality versus efficiency, such as single-linkage, complete-linkage, all-pairs linkage and centroid-linkage measures[1]. These measures are outlined as follows.

**Single linkage** Dissimilarity between two clusters is the shortest distance between any pair of points in the two clusters, i.e., $Dis(C_i, C_j) = \min_{x \in C_i y \in C_j} dist(x, y)$.

**Complete linkage** Dissimilarity between two clusters is the largest distance between any pair of points in the two clusters, i.e., $Dis(C_i, C_j) = \max_{x \in C_i y \in C_j} dist(x, y)$.

**All-pairs linkage** Dissimilarity between two clusters is the average over all pairs of points in the two clusters, i.e., $Dis(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i y \in C_j} dist(x, y)$.

**Centroid linkage** Dissimilarity between two clusters is the distance between centroids in the two clusters, i.e., $Dis(C_i, C_j) = dist(c_i, c_j)$, where $c_i$ and $c_j$ are the cluster centroids of $C_i$ and $C_j$.

Single-linkage clustering generally outputs elongated-shaped clusters, but complete-linkage clustering generally detects compact-shaped clusters. However, both of these are sensitive to noise and outliers. In contrast, all-pairs linkage clustering and centroid-linkage clustering tolerate noise and outliers, but they tend to find globular clusters [1].

#### Divisive Methods

Divisive methods follow a top-down fashion for determining a partitioned structure. They group all points into a single cluster in the first step, and then divide it into several smaller subclusters recursively until only one point is left in a cluster or the dissimilarity between points in a cluster is "small enough" [29]. When building a hierarchical structure, divisive methods allow different trade-offs in the balancing of the node depths and node sizes.

### 2.1.3 Probabilistic-model based Methods

Probabilistic-model based methods provide a general framework that enables each point to belong to multiple clusters in a probabilistic manner. They attempt to optimise the fit between the points and a parametric model based on the assumption that all points are generated from a mixture of underlying probability distributions [1]. A cluster can be defined by a parametric probability distribution, such as a multivariate Gaussian or a multivariate Poisson distribution. A mixture model assumes that all observed points are a mixture of data points from several probabilistic clusters. The clustering problem of this kind of clustering approach is to estimate the parameters of each component distribution for each cluster which best describe the set of observed points.

Given a dataset $D$ with $n$ points $x \in R^d$ and $k$ as the user-specified number of clusters, the task of probabilistic-model based clustering is to infer $k$ probabilistic clusters $C_1, C_2, ..., C_k$ which are most likely to generate the dataset $D$. Let each cluster $C_j (1 \leq j \leq k)$ be associated with a probability $P(C_j) = \rho_j$ which determines the frequency of cluster $j$, and let $\sum_{j=1}^{k} \rho_j = 1$ which ensures all points are generated

by the $k$ clusters. Furthermore, let $f_1, f_2, ..., f_k$ be the probability density functions of clusters $C_1, C_2, ...C_k$, respectively. For a point $x \in D$ the probability that $x$ is generated by cluster $C_j (1 \leq j \leq k)$ is given by $P(x, C_j) = P(C_j)P(x|C_j) = \rho_j f_j(x)$. Thus, the probability of $x$ generated by the set of $k$ clusters $\mathbf{C}$ can be defined as

$$P(x) = \sum_{j=1}^{k} P(x, C_j) = \sum_{j=1}^{k} \rho_j f_j(x) \tag{2.2}$$

Because each data point $x \in D$ is generated independently, we have the joint distribution as

$$P(D) = \prod_{i=1}^{n} P(x_i) = \prod_{i=1}^{n} \sum_{j=1}^{k} \rho_j f_j(x_i) \tag{2.3}$$

Therefore, the objective of this clustering is to find $k$ clusters such that $P(D|\mathbf{C})$ is maximised. The probability density function of a cluster can be arbitrary and the mixtures can be constructed with any type of distribution. In practice, these methods generally assume that the probability density functions are parameterised distributions in order to make the clustering process computationally feasible.

Let $\Theta = \Theta_1, ..., \Theta_k$ be the parameters of the $k$ distributions, respectively. Then Equation 2.3 can be rewritten in the form

$$P(D|\Theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} \rho_j P_j(x_i|\Theta_j) \tag{2.4}$$

where $P_j(x_i|\Theta_j)$ is the probability that $x_i$ is generated from the $j$th distribution/cluster using parameter $\Theta_j$.

The task of probabilistic-model based clustering is to infer different parameters $\Theta$ for maximising Equation 2.4. In statistics, maximum likelihood estimation [18] is one of the most popular methods for parameter estimation. In this case, it considers the best estimate for the parameters as the one that maximises the probability of generating all the observed points, defined as

$$\Theta_{ML} = \arg\max_{\Theta} \{P(D|\Theta)\} \tag{2.5}$$

There exist many different parametric families of distributions. Here I use the Gaussian mixture model, the most popular mixture model, as an example, assuming that each cluster $C_j$ is generated by a 1-dimensional Gaussian distribution with different means $\mu_j$ and variances $\sigma_j$. Then each data point $x \in D$ is generated with the following probability as

$$P(x|\Theta) = \sum_{j=1}^{k} \frac{\rho_j}{\sqrt{2\pi}\sigma_j} e^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}} \tag{2.6}$$

Then the probability of generating all the observed points can be rewritten as

$$P(D|\Theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} \frac{\rho_j}{\sqrt{2\pi}\sigma_j} e^{-\frac{(x_i-\mu_j)^2}{2\sigma_j^2}} \tag{2.7}$$

Then inferring $\Theta$ to maximise Equation 2.7 is the goal of probabilistic-model based clustering when using a univariate Gaussian mixture model. To estimate these parameters, an expectation-maximisation (EM) algorithm can be applied for finding maximum likelihood solutions [18].

### 2.1.4 Density-based Methods

Traditional partitioning and hierarchical methods usually cannot find clusters with arbitrary shapes [1]. Probabilistic-model based methods are parametric, and are effective only if the distribution of the data is known and the cluster shapes follow a particular distribution such as Gaussian [54]. However, cluster shape in real datasets can be arbitrary, which can make these algorithms less effective for discovering all types of clusters. Alternatively, density-based clustering algorithms are non-parametric. They define clusters as regions of high density, which are separated by regions of low density. The advantages of density-based clustering algorithms include discovering clusters of different sizes and shapes while being robust to noise.

### DBSCAN and its Extensions

DBSCAN [22] and DENCLUE [31] are the two most representative algorithms of density-based clustering. They first identify dense regions using a density estimator and then link neighbouring dense regions to form clusters. As such, they can identify arbitrarily shaped clusters. DBSCAN defines the density of a point as the number of points from the dataset that lie in its $\varepsilon$-neighbourhood. DBSCAN labels a point as "core" if its density is higher than a threshold $MinPts$. A point $p$ is directly density-reachable from $q$ if $p$ lies in the $\varepsilon$-neighbourhood of a core point $q$. DBSCAN begins with a core point and links all directly density-reachable points together to form a cluster, until all core points are assigned to some clusters. If a point is neither a core point nor a "boundary" point (i.e., a point which is not a core point but is density-reachable from a core point), then it is considered to be "noise". DENCLUE uses the Gaussian kernel function for density estimation and applies a hill-climbing procedure to link neighbourhood points with high densities. Although DBSCAN and DENCLUE can detect clusters with varied sizes and shapes, they can fail to detect all clusters with widely differing densities [21].

Figure 2.2 shows an example of DBSCAN when $MinPts = 4$. The red, yellow and blue points indicate the core, boundary and noise points, respectively. Density-connected points are linked together to form a cluster.

There are variants of DBSCAN which still make use of density estimation as the key tool to identify clusters with varied densities. Most of them assume that the points inside each cluster are uniformly distributed and thus each cluster can be extracted with a single density threshold. For example, DDSC [11] and EDBSCAN [65] only link neighbouring core points with similar densities together to form clusters. In a real-world dataset in which the data of each cluster is not uniformly distributed, they may split one cluster into a number of unnecessary subclusters. Using a slightly different approach, VDBSCAN [52] uses the same $MinPts$ and selects different $\varepsilon$ values, and then runs DBSCAN several times to detect clusters with different densities. VDBSCAN needs to calculate and sort $k$-dist (the $k^{th}$-nearest neighbour distance, $k = MinPts - 1$) for each point, and it manually identifies sharp changes in the sorted

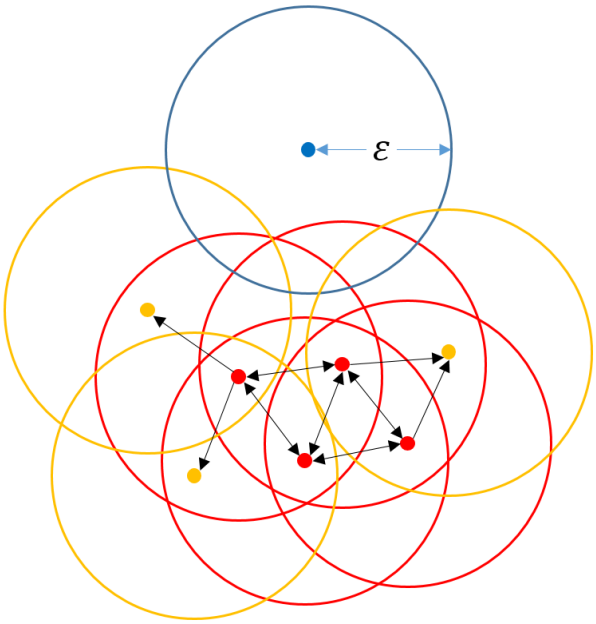


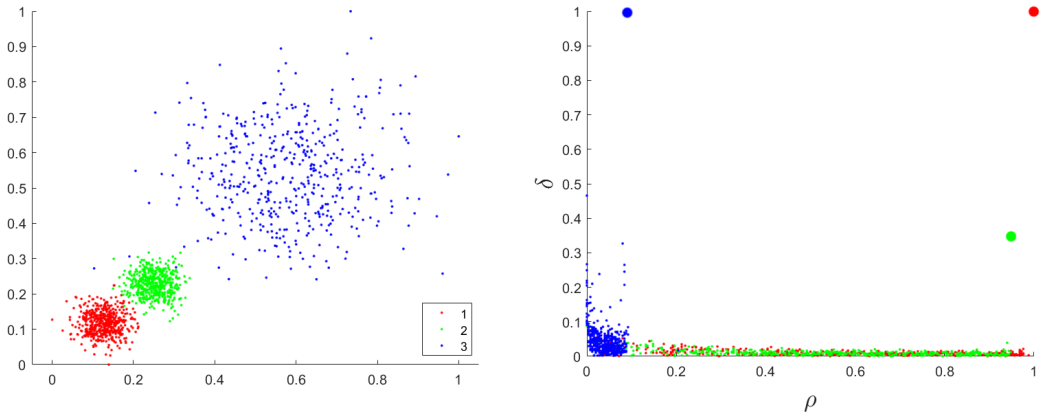

Fig. 2.2 An illustration of the DBSCAN procedure.

$k$-dist plot in order to determine the different $\varepsilon$ values. In practice, it can be difficult to reliably identify these changes since an inappropriate choice of $k$ can seriously affect the plot [74].

Unlike DBSCAN and DENCLUE, which use a global density threshold to identify dense regions and link neighbouring dense regions to form clusters, the density peaks (DP)–based clustering algorithm [66] identifies cluster centres which have local maximum density and are well separated, and then assigns each remaining point to its cluster centre via a linking scheme. It assumes that cluster centres are located at the modes of the estimated density while being sufficiently separated from each other. This clustering process has three steps as follows. For each point, DP firstly calculates its density value $\rho$ using an $\varepsilon$-neighbourhood density estimator, and the minimum distance $\delta$ between it and another point with a higher density value. DP plots a decision graph for all points where the $y$-axis is $\rho \times \delta$, sorted in descending order in the $x$-axis. The points with the highest $\rho \times \delta$ (i.e., high density values and relatively high minimum-distance values) are selected as the cluster centres. After that, each remaining point is connected to its nearest neighbour of higher density, and the points connected or transitively connected to the same cluster centre are grouped into

the same cluster. Once every point is assigned to the corresponding cluster, it finds a border region for each cluster and records the point with the highest density value $\rho_b$ within the border region.[1] Finally, all points of the cluster having density values less than $\rho_b$ are classified as noise.

An example of a decision graph generated by the DP-based clustering algorithm for a dataset is shown is Figure 2.3. The data points with the highest $\rho \times \delta$ in Figure 2.3(b) indicate the three cluster centres. They can be selected for connecting the remaining data points.




(a) Data point distribution (b) Decision graph for the data in (a)

Fig. 2.3 A decision graph generated by the DP-based clustering algorithm [66]. Each colour indicates one of the three clusters.

OPTICS [5] employs a significantly different approach by drawing a "reachability" plot based on $k^{th}$-nearest neighbour distance which reveals the clusters in different regions with respect to the local densities. The plot is produced by a special linear order of all points where spatially adjacent points follow close to each other such that point $p_i$ is the closest to $p_{i-1}$ in terms of the "reachability distance"[2] and the first point $p_0$ is chosen randomly. In addition, it records the reachability distance for each point. Then it draws a reachability plot which has all points ordered in the linear order in the $x$-axis and the reachability distance in the $y$-axis. Because a cluster centre normally has

[1] A border region of a cluster is the set of points which is assigned to that cluster but located within the $\varepsilon$-neighbourhood of a point belonging to another cluster.

[2] The reachability distance of point $q$ to point $p$ is the greater of the "core distance" of $p$ and the distance between $p$ and $q$. The core distance of $p$ is the minimum $\varepsilon$ which makes $p$ a core point (the distance to its $k^{th}$-nearest neighbour, $k = MinPts - 1$).

a higher density or lower reachability distance than the cluster boundary, each cluster is visible as a "valley" in this plot and the clusters can be extracted by a hierarchical method. Although OPTICS only needs one parameter $MinPts$ to draw a reachability plot, the clustering performance depends on the hierarchical method employed.

Figure 2.4 is the reachability plot for a dataset with three Gaussians. Each valley in this plot represents one cluster.

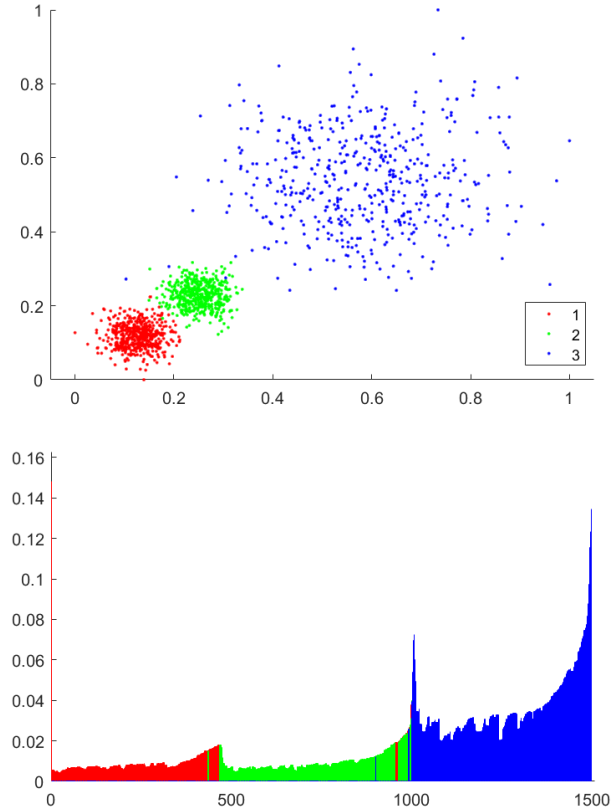



Fig. 2.4 The reachability plot of a dataset with three Gaussians ($MinPts = 10$).

The shared nearest neighbours (SNN) density-based clustering algorithm [21] is another approach to overcoming the weakness of detecting clusters with varied densities. It uses SNN similarity [34] in place of the distance measure in DBSCAN. The SNN similarity of any two data points is the number of shared neighbours if the two point are on each other's nearest-neighbour lists. Since points normally have lower SNN similarity in transition regions between clusters, SNN-based clustering can find clusters with relatively uniform regions with respect to their surroundings regardless of

the density values. However, studies [17, 28, 51] have shown that $k$-nearest neighbour-based algorithms are highly sensitive to the setting of the $k$ parameter.

Figure 2.5 shows the $k$-nearest neighbours of two points $a$ and $b$. The SNN similarity between $a$ and $b$ is 2.

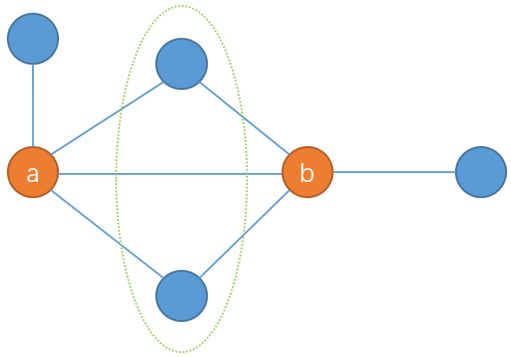



Fig. 2.5 The $k$-nearest neighbours of two points $a$ and $b$ ($k = 4$). They share 2 points in their neighbour lists.

## 2.2 Subspace Clustering Algorithms

As the dimensionality of data increases, traditional "full-space clustering" algorithms become ineffective because the data distribution is invariably sparse in high-dimensional spaces, making density estimates unreliable and the concept of "nearest neighbours" meaningless since all data points are almost equally distant from one another. This problem is referred to as the "curse of dimensionality" [24]. This effect can occur in a dataset with as few as 10–15 dimensions [8]. A high-dimensional dataset is traditionally described by 10 or more attributes, because most clustering algorithms based on traditional distance measures cannot effectively deal with this [29].

Dimensionality-reduction methods are often used to cluster high-dimensional datasets. These methods construct a new lower dimensional space and then perform clustering in the reduced space. However, a global dimensionality-reduction method such as principal components analysis (PCA) [4] is unsuitable when different clusters have different relevant attributes. Figure 2.6 shows an example of a three-dimensional dataset with three subspace clusters.

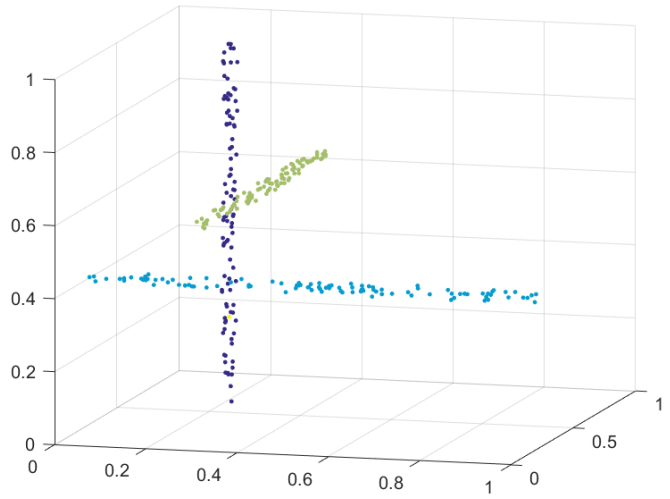



Fig. 2.6 Three subspace clusters exist in a 3-dimensional dataset. Each cluster has two relevant attributes and one irrelevant attribute.

Subspace clustering aims to discover clusters which exist in different subspaces. It works by employing both subspace search and clustering procedures. The key task of clustering in subspaces is to develop appropriate subspace search heuristics [1]. There are two basic techniques for subspace search, namely top-down search [2] and bottom-up search [3]. Different subspace clustering algorithms have been proposed based on these two search directions [68, 41, 60, 57, 78]. Search strategies can be further subdivided into systematic and non-systematic search techniques.

In the following sections, I review different subspace clustering algorithms according to their subspace search strategies and cluster definitions. First, I discuss their properties in terms of five aspects, described as follows:

- **Cluster shape** The ability to identify clusters with arbitrary shapes in different subspaces
- **Non-overlapping clusters** The ability to assign each data point to only one cluster
- **Adaptive density threshold** The ability to set different density thresholds to identify clusters in different dimensional subspaces
- **Handling noise** The ability to accurately identify clusters from a dataset containing noise and outliers

• **Parameter** Whether parameters are easy to set or not

### 2.2.1 Systematic Subspace Search

Systematic search strategies for subspace clustering can be top down [2] or bottom up [3]. Figure 2.7 presents the search space lattice for an example containing four candidate attributes $\{a,b,c,d\}$. Searching all possible subspaces generates $2^d - 2$ candidates (where $d$ is the number of distinct attributes). To avoid having to perform an exhaustive search, the two systematic search strategies utilise certain criteria to prune the search space.

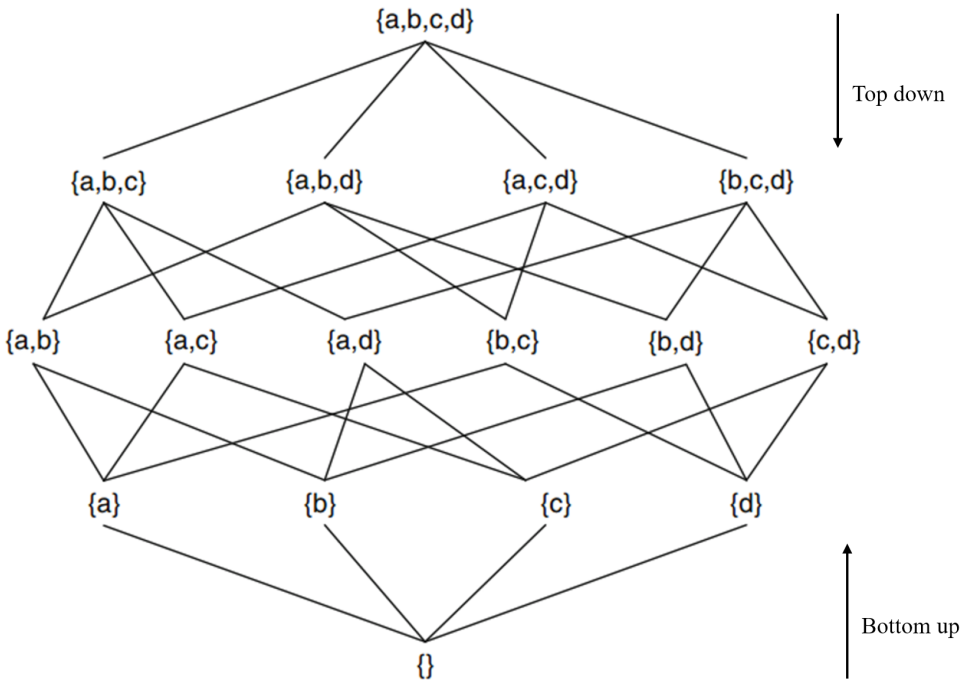



Fig. 2.7 The lattice of candidate subspaces from attributes $\{a,b,c,d\}$.

#### Bottom-up Methods

The bottom-up subspace search strategy uses the *Apriori* algorithm [68]. It starts from all one-dimensional subspaces and then searches for higher dimensional subspaces progressively. This strategy relies on the anti-monotonicity property to prune the search space: if a candidate subspace in a lower dimensional space has no clusters

or is filtered out according to some other criteria, then its projection onto a higher dimensional space will not be traversed.[3]

CLIQUE (Clustering In QUEst) [3] was the first subspace clustering algorithm using a bottom-up subspace search strategy, and relies on the anti-monotonicity property for finding subspace clusters. CLIQUE uses a grid-based structure to investigate dense units at $k-1$ dimensional subspace to produce the possible set of $k$-dimensional units that might contain dense units. It uses a global density threshold to identify dense units. Adjacent dense units are merged to become a single cluster. Figure 2.8 shows an example of a cluster identified by CLIQUE in a two-dimensional dataset.

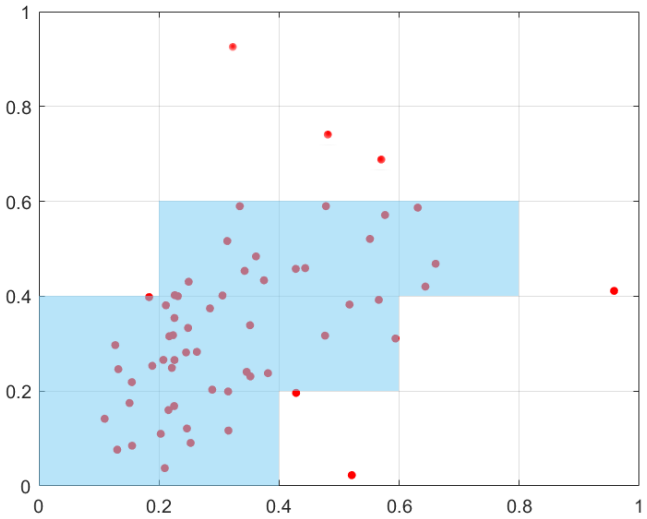



Fig. 2.8 Illustration of a cluster identified by CLIQUE in a two-dimensional dataset.

MAFIA [26] is an extension of CLIQUE that uses a variable-width grid-based density estimator in each subspace in order to improve the efficiency and quality of the clusters found. ENCLUS [14] is similar to CLIQUE but it uses entropy rather than density to measure the "clusterability" of each subspace, since subspaces with lower entropy normally contain more meaningful clusters. ENCLUS uses the same bottom-up strategy of CLIQUE for finding subspace clusters.

[3] To be precise, the anti-monotonicity property is a property of density-based clustering approaches only, where points can be considered either "high density and therefore within a cluster" or "low density and therefore noise". Since density monotonically reduces with increased dimension, if a fixed density threshold is used to define points as high density then the property will be anti-monotone (in the set of attributes), i.e., a low-density point in a subspace with a set of attributes is still a low-density point in a higher dimensional subspace which contains these attributes.

P3C [56] is a grid-based algorithm relying on statistical significance tests to detect subspace clusters. P3C defines significant regions as regions containing significantly more points than expected under a uniform-distribution assumption over a particular subspace. P3C first detects significant regions in individual dimensions by starting with intervals and merging adjacent significant regions together to form cluster cores. It then searches for higher dimensional regions using a bottom-up search strategy. Finally, P3C produces a matrix that records the probability of each point belonging to each cluster. P3C does not output all clusters in all subspaces. It only reports the results of the final clustering using an EM-like clustering procedure [18]. The advantage is that P3C can be used for both numerical and categorical data. Like other density-based algorithms, P3C can detect arbitrarily shaped clusters and is robust to noise as well. The main disadvantage of P3C is that its performance depends on the detected one-dimensional clusters. If clusters have low density in their one-dimensional projections, there will not be enough statistical evidence for further aggregation.

Although the grid-based algorithms provide efficient clustering models, they cannot be applied to high-dimensional datasets, as the number of cells in the grid grows exponentially with the dimensionality of data. The cluster shape detected by grid-based clustering is a polygon (which can be an arbitrary shape for a high grid granularity) in the corresponding subspace where the space is divided in cells by a grid. Their accuracy and efficiency depend on the granularity and the positioning of the grid [1].

In addition, SUBCLU (density-connected Subspace Clustering) [37] and FIRES (FIlter REfinement Subspace clustering) [40] are similar to CLIQUE, but apply different methods for clustering points. SUBCLU applies DBSCAN rather than grid-based clustering on each subspace candidate and outputs all detected clusters when searching for higher dimensional subspaces progressively. SUBCLU provides a better clustering quality but has a higher computational complexity than grid-based algorithms.

FIRES is a general framework for efficient subspace clustering. It runs a clustering algorithm (DBSCAN, SNN, *k*-means or grid-based clustering algorithm) on individual attributes separately in order to find "base clusters" and then utilises an approximation method for merging base clusters iteratively in order to find maximal attribute combi-

nations in high-dimensional subspaces. The similarity between any base clusters is the number of intersecting points. After that, some methods can be used to refine the final clustering results in the refinement step. FIRES use a bottom-up search strategy with a heuristic method for pruning. FIRES scales polynomially in the dimensionality of the dataset. However, it requires parameters for clustering base clusters, and additional three parameters for merging base clusters. Furthermore, its performance is highly dependent on the efficiency of the pruning and refinement steps [40].

Generally, clustering algorithms based on bottom-up searches normally result in redundant clusters, such that a point can belong to many clusters simultaneously, making the clusters difficult to interpret. SUBCLU and the grid-based algorithms use a global density threshold, which leads to a bias against higher dimensional subspaces, i.e., a tighter threshold can filter noise well in low-dimensional subspaces but will lose clusters in higher dimensional subspaces. P3C can avoid this bias by using a threshold based on a statistical significance test, which can be treated as an adaptive density threshold. However, these bottom-up algorithms will take an extremely long time to discover subspace clusters if those clusters are high dimensional, i.e., if the subspace they occupy spans many or most of the original attributes.

#### Top-down Methods

Top-down subspace clustering methods were developed to deal with the issues of redundancy of bottom-up methods. In order to find the best non-overlapping partitioning of the data into subspaces, most top-down methods aim to group points such that each point belongs to only one cluster and each cluster is assigned to a specific subspace [2]. This search method determines the subspace of a cluster starting from the full-dimensional space and then removes the irrelevant attributes iteratively. These methods usually evaluate each attribute for each point or cluster based on the "locality assumption" [41], i.e., the subspace of each cluster can be learnt from the local neighbourhood of cluster members. Note that in some papers, subspace clustering with a top-down method is also called "projected clustering" [1].

The first approach introducing top-down subspace clustering was PROCLUS (PROjected CLUStering) [2], which is a $k$-medoids type [38] clustering algorithm. PROCLUS first randomly selects $k$ potential cluster centres from the data sample and then iteratively refines them. In each loop of the iterative process, for each medoid a set of dimensions is chosen by minimising the standard deviation of the distances of all points in the neighbourhood of the medoids to the corresponding medoid along every dimension. Points are then regrouped to their closest medoid based on the identified subspace of each medoid. In the final step of each loop, medoids are updated based on the points currently assigned to them. The iterative process continues until the clustering quality cannot be improved. The points which are too far away from their closest medoids are identified as noise. The output is $k$ groups of non-overlapping points and a set of noise points. Each cluster has a globular shape in its identified subspace. The performance of PROCLUS is not stable since the initial medoids are randomly selected and are sensitive to irrelevant attributes in the full-dimensional space.

PreDeCon (subspace Preference weighted Density Connected clustering) [10] is a top-down clustering algorithm which relies on DBSCAN [22]. For each point, this algorithm calculates a specialised subspace distance based on the concept of subspace preference, using it for density estimation and linking points. An attribute is relevant to the "subspace preference" of a point if the variance of data in the point's $\varepsilon$-neighbourhood is considered smaller than a threshold. PreDeCon links neighbouring points based on their density distribution on the weighted attributes. The downside of this approach is that there are three parameters that need to be manually set for density estimation [10]. Since it does not identify the subspaces where clusters exist but only weights attributes for each cluster, it is referred to as a "soft" subspace clustering algorithm (while others are called "hard" subspace clustering algorithms). Figure 2.9 shows an example of a cluster identified by PreDeCon in a two-dimensional dataset.

DOC (Density-based Optimal projective Clustering) [64] mixes the bottom-up approach for cluster detection and the top-down approach for iterative improvement in

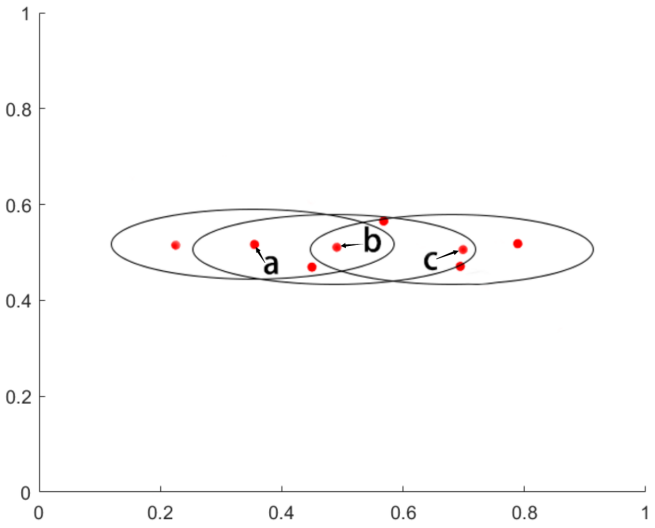



Fig. 2.9 Illustration of how points $a$ and $c$ are weighted connected via $b$ by PreDeCon. The $\varepsilon$-neighbourhood of each point exhibits low variance along an attribute which has a higher weight. Note that the $\varepsilon$-neighbourhood of each point in DBSCAN is a ball without variance along any attribute.

order to reduce redundancy. It uses a fixed density threshold with fixed side-length hypercubes to identify clusters.

### 2.2.2 Non-systematic Subspace Search

There are some subspace clustering algorithms which do not rely on a systematic search strategy but pursue heuristic approaches. STATPC [55] formulates the search for statistically significant regions as subspace clusters with a grid-based structure. It defines the search as an optimisation problem and proposes a heuristic solution to avoid an exhaustive search.

In addition, there is rich literature on soft subspace clustering derived from $k$-means type [30] clustering using similar ideas to PROCLUS, e.g., LAC (Locally Adaptive Clustering) [20], EWKM (Entropy Weighted $k$-Means) [35] and FG-$k$-means (Feature Grouping $k$-Means) [13]. Different from selecting subspaces for each cluster, these kind of algorithms learn weight vectors over attributes with different objective functions and optimisation models, and then incorporate these vectors for distance calculation for the $k$-means algorithm. The distance measure in these algorithms is based on different weights $0 < w_i \leq 1$ for attribute $i = 1, \ldots, d$ as

$$dist_p(x,y) = \sqrt[p]{\sum_{i=1}^{d} w_i \times |x_i - y_i|^p} \tag{2.8}$$

LAC, for example, starts with $k$ centroids and $k$ sets of $d$ weights, and approximates a set of $k$ Gaussians by adapting the weights. Soft subspace clustering algorithms can be seen as some type of normalisation of attributes per cluster. Therefore, these algorithms can detect clusters residing in the skewed but still full-dimensional space with shapes of axis-parallel ellipsoids. In addition, $k$-means–based soft subspace clustering usually requires one parameter $k$ as the number of clusters while other parameters can be set to default values [13].

## 2.3 Measuring Clustering Quality

It is necessary to find a method to evaluate whether a clustering result is accurate. There are many validation measures, focusing on different aspects. They can be categorised into two main types, i.e., internal clustering validation and external clustering validation, depending on whether or not external information such as ground truth is available.

### 2.3.1 Internal Clustering Validation

Internal clustering validation assesses the clustering quality without access to the ground truth for a dataset (i.e., the true assignment of data points to clusters). These validation methods are instead defined based on properties of the cluster definition. For $k$-means related clustering, internal evaluation measures generally examine the compactness and separation of the clusters [1], described as follows.

**Compactness** This measures how closely related the points in a cluster are. A number of measures evaluate cluster compactness based on variance or distance. Lower variance or average pairwise distance indicates better compactness.

**Separation** This measures how separated a cluster is from other clusters. There are some methods for measuring the pairwise distances between cluster centres and pairwise distances between points from different clusters. A higher distance between cluster centres indicates better separation.

The silhouette coefficient [67] is one of the widely used internal clustering validation measures, for a dataset $D$ with $n$ points which is partitioned into $k$ clusters $C_1, ..., C_k$. For each point $x \in D$, let $a(x)$ be the average distance between $x$ and all other points in the cluster to which $x$ belongs, and let $b(x)$ be the minimum average distance from $x$ to all clusters to which $x$ does not belong. $a(x)$ and $b(x)$ measure the compactness and separation, respectively. Given a data point $x \in C_i$ $(1 \leq i \leq k)$, $a(x)$ and $b(x)$ are defined as

$$a(x) = \frac{\sum_{y \in C_i, y \neq x} dist(x,y)}{|C_i| - 1} \tag{2.9}$$

$$b(x) = \min_{C_j : 1 \leq j \leq k, j \neq i} \{ \frac{\sum_{y \in C_j} dist(x,y)}{|C_j|} \} \tag{2.10}$$

The silhouette coefficient of $x$ is then defined as

$$s(x) = \frac{b(x) - a(x)}{max\{a(x), b(x)\}} \tag{2.11}$$

Smaller values of $a(x)$ and larger $b(x)$ make a higher silhouette coefficient score $s(x)$. The average silhouette coefficient value of all points is used to measure a clustering result and a higher value means a better clustering result. However, the silhouette coefficient works well only for convex clusters; it cannot handle a dataset with clusters of arbitrary shapes [1].

### 2.3.2 External Clustering Validation

When external information about ground truth is available, we can compare the ground truth with a clustering result to evaluate the clustering result. The task of this kind of

measure is to assign a score $Q(\Omega, C)$ to a given clustering result $\Omega$ and ground truth $C$. These measures can be computed using the contingency matrix as follows.

Given a dataset $D$ with $n$ points which is partitioned into $k$ clusters $\{\Omega_1, ..., \Omega_k\}$ by a clustering algorithm, where $\bigcup_{i=1}^{k} \Omega_i = D$ and $\Omega_i \bigcap \Omega_j = \emptyset$ for $1 \leq i \neq j \leq k$, let the dataset be partitioned into $k'$ classes $\{ C_1, ..., C_{k'} \}$ according to the ground truth, where $\bigcup_{i=1}^{k'} C_i = D$ and $C_i \bigcap C_j = \emptyset$ for $1 \leq i \neq j \leq k'$. The information overlap between the two partitions can be displayed via a contingency matrix as shown in Table 2.1, where $n_{ij}$ denotes the number of points in cluster $\Omega_i$ from class $C_i$.

Table 2.1 A contingency matrix.

| | $C_1$ | $C_2$ | ... | $C_{k'}$ | $\sum$ |
|---|---|---|---|---|---|
| $\Omega_1$ | $n_{11}$ | $n_{12}$ | ... | $n_{1k'}$ | $n_{1.}$ |
| $\Omega_2$ | $n_{21}$ | $n_{22}$ | ... | $n_{2k'}$ | $n_{2.}$ |
| . | . | . | ... | . | . |
| $\Omega_k$ | $n_{k1}$ | $n_{k2}$ | ... | $n_{kk'}$ | $n_{k.}$ |
| $\sum$ | $n_{.1}$ | $n_{.2}$ | ... | $n_{.k'}$ | $n$ |

Purity is one of the simplest external measures. To compute purity, we first assign each cluster to the true class that is most frequent in the cluster, and then count the number of correctly assigned points and divide by $n$ (the total number of data points). Purity is formally defined as

$$Purity(\Omega, C) = \frac{1}{n} \sum_{i}^{k} \max_{j} n_{ij} \tag{2.12}$$

However, increasing the number of clusters will result in a higher purity score. This is one weakness of this score.

One measure that can overcome this weakness is normalised mutual information (NMI) [69]. It is based on information theory and defined as

$$NMI(\Omega, C) = \frac{I(P;C)}{(H(P) + H(C))/2} \tag{2.13}$$

where the $I$ is mutual information function, defined as

$$I(\Omega;C) = \sum_i \sum_j \frac{n_{ij}}{n} log \frac{n \times n_{ij}}{n_{i.} \times n_{.j}} \tag{2.14}$$

and $H$ is entropy defined as

$$H(\Omega) = -\sum_i \frac{n_{i.}}{n} log \frac{n_{i.}}{n} \tag{2.15}$$

Although entropy and mutual information can be used independently for evaluation of a clustering result, they still have the same problem as purity. NMI fixes this problem by using the combined normalisation.

The F-measure is also a commonly used evaluation method for partitional clustering. It combines the concepts of precision and recall which were developed by the information-retrieval community [1]. The F-measure of a cluster $\Omega_i$ computed with respect to a class $C_i$ is defined as

$$F_{ij} = \frac{2Precision_{ij} \times Recall_{ij}}{Precision_{ij} + Recall_{ij}} \tag{2.16}$$

where $Precision_{ij} = n_{ij}/n_{i.}$ and $Recall_{ij} = n_{ij}/n_{.j}$.

The micro-averaged F-measure for the clustering result is defined as

$$F(\Omega,C) = \sum_{i=1}^{k} \frac{n_{i.}}{n} \max_j \{F_{ij}\} \tag{2.17}$$

and the macro-averaged F-measure is defined as

$$F(\Omega,C) = \frac{1}{k} \sum_{i=1}^{k} \max_j \{F_{ij}\} \tag{2.18}$$

It is worth noting that density-based clustering algorithms may assign some points to noise (unassigned points for which there is no cluster or corresponding class). Because these points cannot be treated as a separate cluster, they cannot be added into the contingency matrix. External measures such as purity and NMI only consider assigned points, i.e., they can have very high scores when many points are unassigned. However, the F-measure can penalise the amount of noise by calculating recall with

$Recall_{ij} = n_{ij}/C_j$. Generally, a macro-averaged F-measure is preferred over a micro-averaged F-measure for clustering validation because it gives equal weight to each cluster.

### 2.3.3 Evaluating Subspace Clustering

There are some external measures which take into account subspace awareness for subspace clustering when the ground truth subspace for each class is available. They measure the correct cluster and the correct relevant dimensions simultaneously [27]. For example, CE (Clustering Error) [62] first uses micro-objects $t(C)$ to represent a cluster $C = (X,A)$ (a set of points $X$ with a set of relevant dimensions $A$), such that $t(C) = \{o_{i,d} \mid o_i \in X \ \wedge \ d \in A\}$. Based on this representation, it maps each detected cluster to each class (ground true cluster) using a contingency matrix and calculates the cardinality of their intersecting micro-objects. Then it uses the Hungarian method [43] for finding the maximum weight matching in a bipartite graph on the contingency matrix. Let $I_{max}$ be the sum over all cardinalities on the matched pair clusters, and let $U$ be the union of the micro-objects of the truth classes and detected clusters; CE can be calculated by

$$CE(\Omega, C) = \frac{|U| - I_{max}}{|U|} \tag{2.19}$$

## 2.4 Summary

There are different kinds of clustering algorithms depending on the specific assumptions regarding the type of clusters being sought and the model used. Density-based clustering algorithms can detect arbitrarily shaped and sized clusters. They are non-parametric, and are tolerant of noise and outliers. DBSCAN, as a representative of density-based clustering, suffers from a failure to detect clusters of varied densities, while existing solutions mitigate this weakness in DBSCAN without knowing the exact condition under which DBSCAN fails. From another perspective, it is instructive to note that some of these solutions rely on the use of a $k$-nearest neighbour

method, either to estimate density or to determine different $\varepsilon$ values, in addition to the $\varepsilon$-neighbourhood density estimator used in DBSCAN. Studies [17, 28, 51] have shown that $k$-nearest neighbour–based algorithms have a highly sensitive $k$ parameter.

In contrast, the proposed approaches in Chapter 3 avoid the use of $k$-nearest neighbour methods or an additional density estimator, and contribute to a less brittle solution.

For subspace clustering on high-dimensional data, Table 2.2 compares properties of different subspace clustering algorithms. Most density-based algorithms (including grid-based) have more than two critical parameters for defining clusters which are very difficult to set in practice. The performance is sensitive to their settings. None of the $k$-medoids and $k$-means type clustering algorithms can detect non-globular clusters, while density-based algorithms can. Although the grid-based methods can accelerate density-based clustering, they cannot scale up to high dimensionality because the number of cells increases exponentially with the increasing number of dimensions.

Existing subspace clustering procedures perform clustering by measuring the similarity between points in the given feature space, and the subspace selection and clustering processes are tightly coupled. In order to easily employ a different cluster definition, it is important to develop an efficient subspace clustering algorithm which explicitly separates the candidate subspace selection and clustering processes into two independent processes. The proposed new subspace clustering framework, named *CSSub* (Shared Subspaces Clustering) and described in Chapter 4, has these desired unique features.

For clustering evaluation, there is no common solution to cluster validation [1]. Internal measures are typically used for clustering algorithms which have specific objective functions [27]. These measures are based on the underlying cluster definition. For density-based clustering algorithms, since there is no objective function, no internal measure is meaningful and can be used for them. External evaluation measures such as purity and NMI only take into account the points assigned to clusters and do not account for noise. A clustering which assigns the majority of the points to noise may result in a high clustering performance. The F-measure is more suitable than

Table 2.2 Properties of subspace clustering algorithms and the proposed *CSSub* framework.

| Algorithm | Subspace search strategy | Cluster type | Arbitrarily shaped clusters | Non-overlapping clusters | Adaptive density threshold | Handling noise | Easy parameter setting | Clustering without original attributes |
|---|---|---|---|---|---|---|---|---|
| CLIQUE | Bottom-up | Grid | √ | | | √ | | |
| MAFIA | Bottom-up | Grid | √ | | | √ | | |
| ENCLUS | Bottom-up | Grid | √ | | | √ | | |
| P3C | Bottom-up | Grid | √ | √ | √ | √ | √ | |
| FIRES | Bottom-up | Flexible | √ | | √ | √ | | |
| SUBCLU | Bottom-up | Density | √ | | | √ | | |
| PROCLUS | Top-down | $k$-medoids | | √ | | √ | √ | |
| PreDeCon | Top-down | Density | √ | √ | | √ | | |
| DOC | Hybrid | Grid | √ | √ | | | | |
| STATPC | Heuristic | Grid | √ | | √ | √ | | |
| LAC | Optimisation | $k$-means | | √ | | | √ | |
| EWKM | Optimisation | $k$-means | | √ | | | √ | |
| FG-k-means | Optimisation | $k$-means | | √ | | | √ | |
| CSSub | Non-search | Flexible | √ | √ | √ | √ | √ | √ |

purity and NMI in assessing the clustering performance of density-based clustering and identifying noise.

For the evaluation of subspace clustering, there are certain evaluation measures based on both point awareness and subspace awareness criteria [62]. However, I cannot apply them to most of my experimental datasets because these measures require ground truth about subspaces, which is not available for many real-world datasets. Because there are ground truth cluster labels for every dataset, I only use the F-measure as a relatively fair clustering evaluation for all experimental evaluation in this project.

# Chapter 3

# Detecting Clusters with Varied Densities

*Variety's the very spice of life, that gives it all its flavour.*

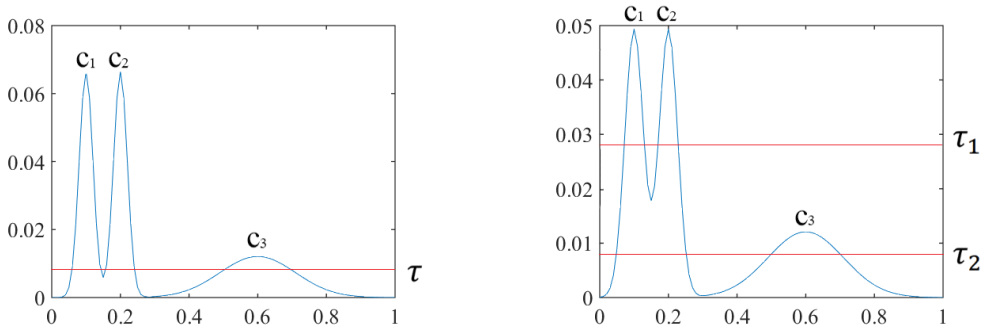
WILLIAM COWPER

Density-based clustering algorithms, such as DBSCAN [22] and DENCLUE [31], are an important class of clustering algorithms that are able to discover clusters of different sizes and shapes while being robust to noise. They define clusters as regions of high densities which are separated by regions of low densities. However, most of these algorithms have difficulty finding clusters with widely varied densities [21]. Only a few of them can find clusters with greatly varied densities and only under limited conditions.

To answer Research Question 1, in this chapter I identify the condition under which most density-based clustering algorithms fail to find all clusters of differing densities and then provide three approaches (ReCon, ReScale and ReMass) to overcome this weakness.

## 3.1 Motivation

Figure 3.1(a) depicts the clustering result of DBSCAN-like algorithms, where the $x$-axis is the point attribute value and the $y$-axis is the density value at that point.




(a) STS (single threshold separable) distribution (b) Non-STS (non-single threshold separable) distribution

Fig. 3.1 (a) A mixture of three Gaussian distributions that can be separated using a single density threshold; (b) a mixture for which no single global density threshold can separate all three clusters.

Notice that there is no problem in identifying all clusters in this dataset even though the difference in density between cluster $C_3$ and the other two clusters is very large. DBSCAN and DENCLUE, however, fail to detect all clusters in a slight variation of the data distribution depicted in Figure 3.1(b), where the peak densities of the different clusters remain the same, but the difference minimum densities between the clusters is increased. The data distribution is a mixture of three univariate Gaussian distributions where the standard deviations of the components are such that the maximum density for cluster $C_3$ is less than the minimum density between clusters $C_1$ and $C_2$. Thus, no single density threshold can be found that separates out all three clusters. Using a high threshold $\tau_1$ results in clusters $C_1$ and $C_2$ being identified but not $C_3$ since all points in that region would be treated as noise. Using a lower threshold $\tau_2$ results in the identification of cluster $C_3$ but all data points in $C_1$ and $C_2$ are assigned to the same (single) cluster.

The exact condition under which density-based clustering algorithms will fail has not been formalised previously, as far as I know. Existing solutions for solving this problem mitigate the weakness in DBSCAN without knowing the exact condition. Because these algorithms do not fail in all datasets having clusters of differing densities, it is important to pin down this condition.

## 3.2 Condition under which Density-based Algorithms Fail

The problem of density-based clustering algorithms can be formalised as follows.

Let $D = \{x_1, x_2, ..., x_n\}$, $x_i \in R^d, x_i \sim F$ denote a dataset of $n$ points each sampled independently from a distribution $F$. Let $\widehat{pdf}(x)$ denote the density estimate at point $x$ used (either explicitly or implicitly) by a particular density-based clustering algorithm which approximates the true density $pdf(x)$. A set of clusters $\{C_1, \ldots, C_\varsigma\}$ is defined as non-empty and non-intersecting subsets: $C_i \subset D, C_i \neq \emptyset, \forall_{i \neq j}\ C_i \cap C_j = \emptyset$. Let $c_i = \arg\max_{x \in C_i} \widehat{pdf}(x)$ denote the mode (point of the highest estimated density) for cluster $C_i$ and $p_i = \widehat{pdf}(c_i)$ denote the corresponding peak density value.

Finally, let $\mathcal{N}_\varepsilon(x)$ be the $\varepsilon$-neighbourhood of $x$, $\mathcal{N}_\varepsilon(x) = \{y \in D \mid dis(x,y) \leqslant \varepsilon\}$, where $dis(\cdot,\cdot)$ is the dissimilarity function ($dis : R^d \times R^d \to R$) used by the density-based clustering algorithm.

**Definition 1.** A **core point** $x \in D$ is a point with estimated density above or equal to a user-specified threshold $\tau$, i.e., $(\widehat{pdf}(x) \geqslant \tau) \leftrightarrow Core^\tau(x)$.

**Definition 2.** A **non-cyclic path** linking points $x_i$ and $x_j$, $path(x_i, x_j)$, is defined as a sequence of unique points starting with $x_i$ and ending with $x_j$ where adjacent points lie in each other's neighbourhood: $(x_{\pi(1)}, x_{\pi(2)}, ..., x_{\pi(k)})$. Here $\pi()$ is a mapping $\pi : \{1, ..., k\} \to \{1, ..., n\}$ such that $\forall_{s \neq t \in \{1,...,k\}}\ (\pi(s) \neq \pi(t)) \wedge (\pi(1) = i) \wedge (\pi(k) = j) \wedge (\forall_{v \in \{1,...,k-1\}}\ x_{\pi(v+1)} \in \mathcal{N}_\varepsilon(x_{\pi(v)}))$.

**Definition 3.** A point $x_i$, which is density-connected with another point $x_j$, is defined as $Connect^\tau_\varepsilon(x_i, x_j) \leftrightarrow \exists_{path(x_i,x_j)}\ \forall_{y \in path(x_i,x_j), y \neq x_i, y \neq x_j}\ Core^\tau(y)$.

**Definition 4.** A cluster detected by a density-based algorithm is a maximal set of density-connected points, i.e., $C_i = \{x \in D \mid Connect^\tau_\varepsilon(x, c_i)\}$.

**Definition 5.** A **noise point** $x \in D$ is a non-core point and is not density-connected with a core point, i.e., $\neg Core^\tau(x) \wedge (\neg \exists_y\ Core^\tau(y) \wedge Connect^\tau_\varepsilon(x,y)) \leftrightarrow Noise^\tau(x)$.

Let $\mathcal{P}_{ij} = \{(x_{\pi(1)}, x_{\pi(2)}, ..., x_{\pi(k)}) \mid x_{\pi(1)} = c_i \wedge x_{\pi(k)} = c_j\}$ denote the set of all non-cyclic paths linking the modes of clusters $C_i$ and $C_j$, and then let $g_{ij} = \max_{path \in \mathcal{P}_{ij}} \min_{x \in path} \widehat{pdf}(x)$ be the largest of the minimum values along any paths in $\mathcal{P}_{ij}$.

Let each cluster $C_i$ be represented by its mode $c_i$ in order for a density-based clustering algorithm to be able to (reliably) identify and separate all clusters in a dataset $D = \{x_1, \ldots, x_n\}$. The condition that the estimated density at the mode of any cluster is greater than the maximum of the minimum estimated density along any path linking any two modes is given as

$$\min_{k \in \{1,...,\varsigma\}} p_k > \max_{i \neq j \in \{1,...,\varsigma\}} g_{ij} \tag{3.1}$$

This condition implies that there must exist a threshold $\tau$ that can be used to break all paths between the modes by assigning regions with estimated density less than $\tau$ to noise, i.e.,

$$\exists_\tau \forall_{k, i \neq j \in \{1,...,\varsigma\}} \; p_k \geqslant \tau > g_{ij}$$

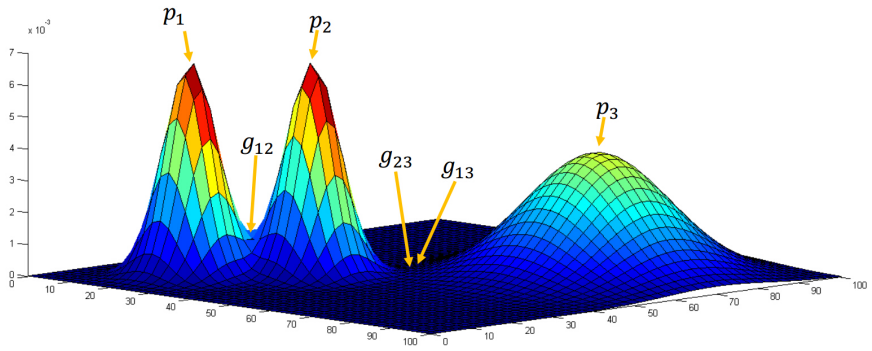



Fig. 3.2 An example of estimated density distribution for a 2-dimensional dataset consisting of three Gaussian distributions.

Figure 3.2 illustrates how to find a threshold $\tau$ to separate all clusters for a two-dimensional dataset consisting of a mixture of three Gaussian distributions. Here $g_{12}$ is the maximum of the minimum estimated density along any path linking cluster $C_1$ and $C_2$, which is larger than $g_{13}$ and $g_{23}$. As $p_3$ is smaller than $p_1$ and $p_2$, thus $\tau$ should be set between $g_{12}$ and $p_3$ in order to identify all three clusters.

Here I provide the theorem on density-based clustering algorithms as follows:

**Theorem 1.** If a density-based clustering algorithm uses a global threshold on the estimated density to identify core points and links neighbouring core points together to form clusters, then the requirement given in Equation (3.1) on the density estimates and cluster definitions provides a necessary condition for the algorithm to be able to identify all clusters.

**Proof 1.** I use a counterexample to prove the necessary condition. Given a dataset consisting of more than one cluster ($\varsigma \geqslant 2$) which violates the condition specified in Equation (3.1), i.e., where $\exists_{k,i\neq j\in\{1,...,\varsigma\}}\ p_k \leqslant g_{ij}$, two situations will occur depending on the value of $\tau$:

(1) $g_{ij} < \tau$: Clusters which have modes less than $\tau$ are treated as noise $\forall_{u\in\{1,...,\varsigma\}}\ (p_u < \tau) \leftrightarrow Noise(c_u)$

(2) $g_{ij} \geqslant \tau$: Let $path^*_{ij}$ be a non-cyclic path corresponding to $g_{ij}$; I have $path^*_{ij} = (x_{\pi(1)},...,x_{\pi(k)})$, $x_{\pi(1)} = c_i$, $x_{\pi(k)} = c_j$ and $g_{ij} = \min\limits_{x\in path^*_{ij}} \widehat{pdf}(x)$. Due to the fact that $\forall_{x\in path^*_{ij}}\ \widehat{pdf}(x) \geqslant g_{ij} \geqslant \tau$, then all points in $path^*_{ij}$ are core points. Since $\forall_{v\in\{1,...,k-1\}}\ x_{\pi(v+1)} \in \mathcal{N}_\varepsilon(x_{\pi(v)})$, any two adjacent points in $path^*_{ij}$ are linked, and then $c_i$ and $c_j$ are transitively linked. So cluster $C_i$ and cluster $C_j$ are merged into one group $C_i = C_j$. Similarly, $\forall_{s\neq t\in\{1,...,\varsigma\}}\ g_{st} \geqslant \tau \rightarrow C_s = C_t$ because all points in a non-cyclic path containing $g_{st}$ are linked.

Therefore, in either situation, not all clusters have been identified. □

**Definition 6.** A data distribution is single threshold separable (STS) distribution if it meets the inequality in Equation (3.1).

Therefore, I define a non-STS distribution as one which violates the inequality in Equation (3.1). This is then the condition under which density-based clustering algorithms fail to identify all clusters.

## 3.3 Approaches for Overcoming the Weakness of Density-based Algorithms

### 3.3.1 Using Density-ratio instead of Density (the ReCon Approach)

In order to detect low-density clusters by DBSCAN, I need to replace the default density measure with a measure that attributes larger values to these clusters. Here I discuss the idea of replacing the density estimate with a density-ratio estimate, where the density is normalised with respect to the average density over the surrounding area. The aim of this method is to find clusters as regions of **high local densities** which are separated by regions of **low local densities**. By doing this, I am able to deal with the variation in (average) density across clusters and thereby allow for a single threshold to be used to identify all clusters in the dataset.

Instead of identifying a core point based on its density, I propose to use the density-ratio, which is a ratio of the density of a point and the average density over its $\eta$-neighbourhood. It is defined as follows

$$rpdf_{\eta}(x) = \frac{pdf(x)}{\widehat{pdf}_{\eta}(x)} \tag{3.2}$$

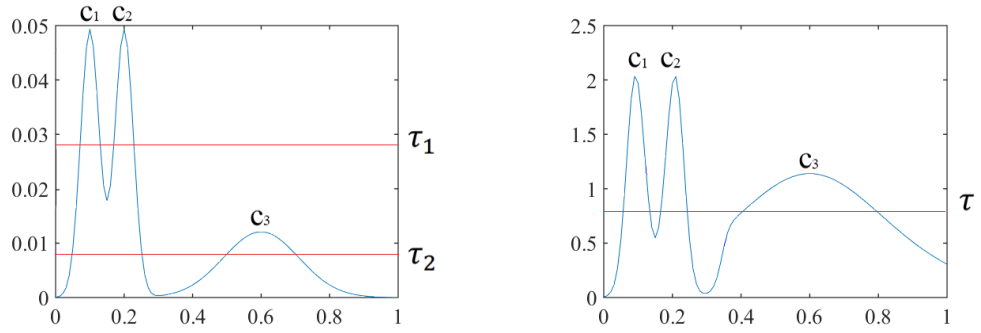
where $\widehat{pdf}_{\eta}(x)$ denotes the average density value over $\eta$-neighbourhood of $x$ estimated by the density estimator in a clustering algorithm.

A core point is now a point whose estimated density-ratio is above a threshold. There are two critical scenarios:

1 If $x$ is at a maximum local density of $\mathcal{N}_{\eta}(x)$, i.e.,$\forall_{y \in \mathcal{N}_{\eta}(x)}\ pdf(x) \geqslant pdf(y)$ then $rpdf_{\eta}(x) \geqslant 1$ since $\widehat{pdf}_{\eta}(x) \leqslant pdf(x)$.

2 If $x$ is at a minimum local density of $\mathcal{N}_{\eta}(x)$, i.e., $\forall_{y \in \mathcal{N}_{\eta}(x)}\ pdf(x) \leqslant pdf(y)$ then $rpdf_{\eta}(x) \leqslant 1$ since $\widehat{pdf}_{\eta}(x) \geqslant pdf(x)$.

If a suitable $\eta$ can be chosen such that (1) for each cluster, the peak density is higher than average density over the $\eta$-neighbourhood around the peak (i.e., $\forall_{k \in \{1,\ldots,\varsigma\}} p_k > \widehat{pdf}_{\eta}(c_k)$) and (2) there exists a point along every path linking two cluster peaks where




(a) Non-STS distribution (b) Density-ratio distribution for (a)

Fig. 3.3 (a) A mixture for which no single global density threshold can separate all three clusters; (b) a replotting of distribution in (a) in terms of density-ratio distribution which is a ratio of the density of a point and the density of its $\eta$-neighbourhood, and allows for a single threshold to be used to separate all three clusters.

the density is lower than the average density over the $\eta$-neighbourhood around the point (i.e., $\forall_{i\neq j\in\{1,\ldots,\varsigma\}}\forall_{path\in\mathcal{P}_{ij}}\ \exists_{x\in path}\ pdf(x) < \widehat{pdf}_\eta(x)$), then

$$\min_{k\in\{1,\ldots,\varsigma\}} \frac{p_k}{\widehat{pdf}_\eta(c_k)} > \max_{i\neq j\in\{1,\ldots,\varsigma\}} \frac{g_{ij}}{\widehat{pdf}_\eta(v_{ij})} \tag{3.3}$$

where $v_{ij}$ is the point corresponding to $g_{ij}$ and $pdf(c_k) = p_k$ and $pdf(v_{ij}) = g_{ij}$.

Having a suitable $\eta$ satisfying the above condition, I would be able to choose a single $\tau$ around 1 to break all paths between different cluster peaks by assigning $v_{ij}$ to noise in the density-ratio distribution. Figure 3.3(b) shows the distribution of density-ratio for the density function shown in Figure 3.3(a), where all clusters can now be identified using a single threshold. Notice that all three clusters now have about the same density-ratio, in contrast to the large density difference between $C_3$ and the other two clusters. This is a result of setting $\eta = 0.1$, enabling the clusters to be identified using a single threshold, which is otherwise impossible in the original density distribution.

I Notice that the efficacy of this technique depends critically on the definition of the neighbourhood. If the neighbourhood is too large, then the density-ratio will approximate the true density and show no advantage, while if the neighbourhood is too small, the density-ratio approaches 1 and provides no information.

In general, $pdf(x)$ can be estimated via a small $\varepsilon$-neighbourhood (as in the case of DBSCAN [22]) as follows

$$\widehat{pdf}_{\varepsilon}(x) = \frac{1}{nV_d^{\varepsilon}} \sum_j \mathbf{1}(\|x - x_j\| \leqslant \varepsilon) \tag{3.4}$$

where $\mathbf{1}()$ denotes the indicator function and $V_d^{\varepsilon} \propto \varepsilon^d$ is the volume of a $d$-dimensional ball of radius $\varepsilon$. Similarly, $\widehat{pdf}_{\eta}(x)$, the average density value over the $\eta$-neighbourhood of $x$, can be estimated in the same way, where $\varepsilon < \eta$.

The density-ratio estimated at $x$ is

$$\widehat{rpdf}_{\varepsilon,\eta}(x) = \frac{\widehat{pdf}_{\varepsilon}(x)}{\widehat{pdf}_{\eta}(x)} = (\frac{\eta}{\varepsilon})^d \frac{\sum_j \mathbf{1}(\|x - x_j\| \leqslant \varepsilon)}{\sum_j \mathbf{1}(\|x - x_j\| \leqslant \eta)} \tag{3.5}$$

When $\varepsilon$ and $\eta$ are chosen properly, $\widehat{rpdf}_{\varepsilon,\eta}(x)$ has the properties of Equation (3.2), i.e., the ability to detect maximum local density points and minimum local density points. Then I can compute the density-ratio of Equation (3.5) using the two densities estimated by the density estimator of any density-based clustering algorithm (from the $\varepsilon$- and $\eta$-neighbourhoods). After the distribution of $\widehat{rpdf}_{\varepsilon,\eta}(x)$ is produced, a threshold between 1 and $(\frac{\eta}{\varepsilon})^d$ can then be used to find all clusters. (Notice that the volume ratio $(\frac{\eta}{\varepsilon})^d$ is constant and can be incorporated into the threshold in the range $[(\frac{\varepsilon}{\eta})^d, 1]$.)

The reconditioned clustering algorithm with the density-ratio is the proposed method and it can now detect all clusters with varied densities using a single threshold.

### 3.3.2 Density-ratio based Scaling (the ReScale Approach)

Instead of reconditioning an existing density-based algorithm, I provide an alternative approach whereby I rescale the dataset in such a way that a density-based clustering algorithm can be applied directly (without modification) to the rescaled dataset to perform density-ratio based clustering as described in the last section. More precisely, I show that the density-ratio based clustering can be achieved by first rescaling the

dataset in such a way that the **estimated density of each rescaled point is approximately the estimated density-ratio of that point in the original space**.

ReScale uses a (one-dimensional) histogram-based density estimation to rescale the data on each dimension according to the cumulative distribution function of the large bandwidth ($\eta$) estimation. The effect of this procedure is to make the data approximately uniform (on each dimension) according to the large bandwidth ($\eta$) density estimation. After transformation, the density-based clustering with a small bandwidth ($\varepsilon$) will approximate the density-ratio based clustering. Here I provide the theoretical justification for rescaling a one-dimensional dataset.

Given a one-dimensional dataset, let $\widehat{cdf}_\eta(x) = \int_{-\infty}^{x} \widehat{pdf}_\eta(x')dx'$ be the cumulative distribution function. The dataset $D$ is rescaled to $S$ using the following mapping function

$$\forall_{x\in D, y\in S} \;\; y = \widehat{cdf}_\eta(x) \tag{3.6}$$

Notice that this is a probability integral transform [63] such that $\forall_{x\in D,y\in S}\ \widehat{pdf}_\eta(y) = \widehat{pdf}_\eta(\widehat{cdf}_\eta(x)) = 1$, which is a uniform distribution.

To see how this transformation of a dataset results in a new dataset whereby the density around a given point approximates the density-ratio around the corresponding point in the original dataset, consider the following. Let $\widehat{pdf}_\varepsilon(y)$ be the estimated density around $y = \widehat{cdf}_\eta(x)$ in the transformed space using a window of size $\varepsilon < \eta$. This estimate can be rewritten in terms of density estimates from the original space as

$$\begin{aligned}
\widehat{pdf}_\varepsilon(y) &= \frac{1}{2n\varepsilon}\sum_j \mathbf{1}(\|y - y_j\| \leqslant \varepsilon) \\
&= \frac{1}{2n\varepsilon}\sum_j \mathbf{1}(\|\widehat{cdf}_\eta(x) - \widehat{cdf}_\eta(x_j)\| \leqslant \varepsilon) \\
&\approx \frac{1}{2n\varepsilon}\sum_j \mathbf{1}(P_{X\sim\widehat{pdf}_\eta}(X \in [\min(x,x_j), \max(x,x_j)]) \leqslant \varepsilon)
\end{aligned}$$

When the data size $n = |D|$ is large, this can be approximated by

$$\begin{aligned}\widehat{pdf}_{\varepsilon}(y) &\approx \frac{1}{2n\varepsilon}\sum_{j}\mathbf{1}(\widehat{pdf}_{\eta}(x) * \|x-x_j\| \leqslant \varepsilon)\\ &\approx \frac{1}{2\varepsilon}P_{X\sim F}(\widehat{pdf}_{\eta}(x) * \|x-X\| \leqslant \varepsilon)\\ &= \frac{1}{2\varepsilon}P_{X\sim F}(\|x-X\| \leqslant \frac{\varepsilon}{\widehat{pdf}_{\eta}(x)})\end{aligned}$$

If $pdf(x)$ varies slowly around $x$, then I can approximate this by

$$\begin{aligned}\widehat{pdf}_{\varepsilon}(y) &\approx \frac{1}{2\varepsilon}\frac{P_{X\sim F}(\|x-X\| \leqslant \varepsilon)}{\widehat{pdf}_{\eta}(x)}\\ &\approx \frac{1}{2n\varepsilon}\frac{\sum_j \mathbf{1}(\|x-x_j\| \leqslant \varepsilon)}{\widehat{pdf}_{\eta}(x)} = \frac{\widehat{pdf}_{\varepsilon}(x)}{\widehat{pdf}_{\eta}(x)}\end{aligned}$$

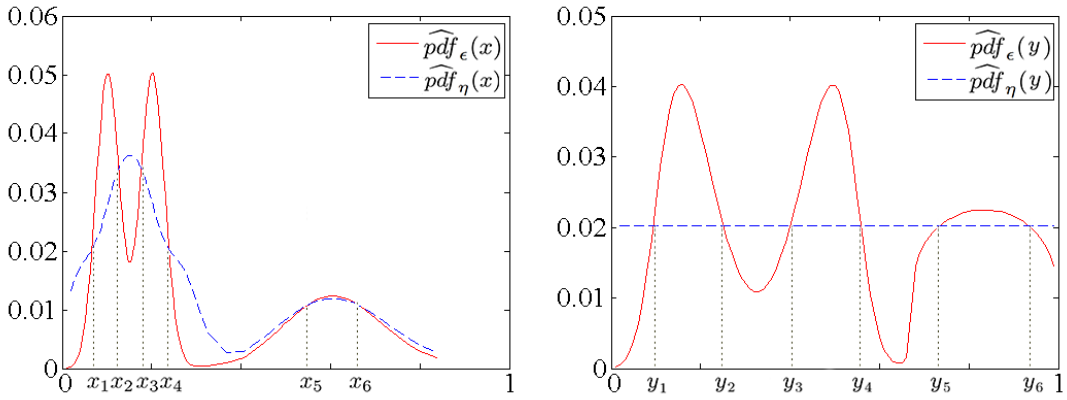
Therefore $\widehat{pdf}_{\varepsilon}(y) \approx \widehat{rpdf}_{\varepsilon,\eta}(x)$, i.e., when applying a density estimator with a small $\varepsilon$ to the rescaled data, the value calculated is approximately the estimated density-ratio of the original data. This is how I will enable an existing density-based clustering algorithm to perform density-ratio based clustering by simply rescaling a given dataset, as shown in Table 3.1.

Table 3.1 Enabling an existing density-based clustering algorithm to perform density-ratio based clustering by simply rescaling $D$ to $S$ using $\widehat{cdf}_{\eta}(\cdot)$.

| Dataset | $D$ (original data) | $S$ (rescaled data) |
|---|---|---|
| Points | $x$ | $y = \widehat{cdf}_{\eta}(x)$ |
| Density estimator | $\widehat{pdf}_{\varepsilon}(x)$ | $\widehat{pdf}_{\varepsilon}(y) \approx \widehat{rpdf}_{\varepsilon,\eta}(x)$ |
| Clustering type | density-based | density-ratio based |

For a one-dimensional dataset $D$, $\widehat{cdf}_{\eta}(x)$ can be estimated using a histogram and the mapping can be implemented as follows: first, split the one-dimensional space into $\iota$ equal intervals with $\iota+1$ boundary positions $\{\chi_1, \chi_2, ..., \chi_{\iota+1}\}$. Second, at each boundary position $\chi_i$ record $\widehat{pdf}_{\eta}(\chi_i) = \frac{1}{2n\eta}\sum_j \mathbf{1}(\|\chi_i - x_j\| \leqslant \eta)$. Last, for every point $x$ in the original dataset $D$, $y$ in $S$ is derived as $\forall_{x\in D}\ y = \sum_{\chi_i \leqslant x}\widehat{pdf}_{\eta}(\chi_i)$. I then apply the *min-max* normalisation so that $y$ is in $[0,1]$.

An example of such a mapping is shown in Figure 3.4, where $\widehat{pdf}_{\varepsilon}(x)$ and $\widehat{pdf}_{\eta}(x)$ for a particular non-STS distribution are shown in Figure 3.4(a). Their mapped distributions $\widehat{pdf}_{\varepsilon}(y)$ and $\widehat{pdf}_{\eta}(y)$ are shown in Figure 3.4(b) using $\varepsilon = 0.01$, $\eta = 0.1$ and $\iota = 10000$. Notice that the cross-cover points $x_1, ..., x_6$ in Figure 3.4(a) are mapped to $y_1, ..., y_6$ in Figure 3.4(b). The mapping yields that the three clusters in the original space (corresponding to the two high-density areas of $\widehat{pdf}_{\eta}(x)$ between $x_1$ and $x_4$ and between $x_5$ and $x_6$ in Figure 3.4(a)) are expanded in the new space in Figure 3.4(b). Notice that the area between $y_1$ and $y_4$ is expanded far more than that between $y_5$ and $y_6$ in Figure 3.4(b) because the former has a higher density than the latter. On the other hand, the largest valley (shown as the area of $\widehat{pdf}_{\eta}(x)$ between $x_4$ and $x_5$ in Figure 3.4(a)) has shrunk significantly in Figure 3.4(b). As a result, the densities at the peaks of different clusters of $\widehat{pdf}_{\varepsilon}(x)$ are closer after this scaling. Notice that the estimated density of $y$ in the $\varepsilon$-neighbourhood in Figure 3.4(b) is approximately the estimated density-ratio of $x$ in Figure 3.3(b), i.e., $\forall_{y \in S, x \in D}\ \widehat{pdf}_{\varepsilon}(y) \approx \widehat{rpdf}_{\varepsilon,\eta}(x)$.




(a) $\widehat{pdf}_{\varepsilon}(x)$ VS $\widehat{pdf}_{\eta}(x)$ (b) $\widehat{pdf}_{\varepsilon}(y)$ VS $\widehat{pdf}_{\eta}(y)$, where $y = \widehat{cdf}_{\eta}(x)$

Fig. 3.4 (a) A plot of $\widehat{pdf}_{\varepsilon}(x)$ and $\widehat{pdf}_{\eta}(x)$; (b) the corresponding plot of $\widehat{pdf}_{\varepsilon}(y)$ and $\widehat{pdf}_{\eta}(y)$, where $x$ is mapping to $y$ via Equation (3.6).

A multi-dimensional dataset can be rescaled based on the one-dimensional mapping method mentioned above by repeated this scaling on each dimension. The assumption of the method is that clusters do not overlap significantly on each attribute projection. If the overlap is significant, then a random projection method can be applied as follows: $D \subset \mathbb{R}^d \rightarrow S \subset \mathbb{R}^t$ with $t$ linear projections, where the one-dimensional scaling method

is applied on each projection to form a new dimension in the new space. Random linear projection is a feasible method to reduce the overlapping effect, and enhances the efficiency of ReScale, if the density distribution on each projection meets the condition in Equation (3.3). Notice that many random linear projections are required ($t$ should be large) in order to obtain a more stable clustering result. An example of the random projection method is provided in Section 6. In all the following experiments, ReScale is implemented using the original attributes as default.

### 3.3.3 A Data-dependent Dissimilarity (the ReMass Approach)

Since most of the density-based algorithms rely on distance computation as their core operation, if a new dissimilarity measure possesses the desired property of inducing a density-ratio measure as specified in the last two subsections, then the existing algorithms can be modified by simply replacing the distance computation with the new dissimilarity computation, leaving the rest of the procedure unchanged.

In order to design such a measure, I first discuss the density estimator $\widehat{pdf}(x)$ implied by the DBSCAN clustering algorithm for the estimation of the density value at point $x$. DBSCAN labels a point $x$ as core if at least $MinPts$ points lie within its $\varepsilon$-neighbourhood (including $x$). This is equivalent to setting a threshold $\tau$ on the density at a point and estimating that density via kernel density estimation as follows

$$\widehat{pdf}(x) = \frac{1}{n}\sum_{j} K_x(x_j) \tag{3.7}$$

where $n$ is the dataset size and the kernel function[1] being used is the uniform distribution over a ball of radius $\varepsilon$, i.e.,

$$K_x(y) = \frac{\mathbf{1}(\|x-y\|_2 \leqslant \varepsilon)}{V_d^{(\varepsilon)}} \tag{3.8}$$

Here $\mathbf{1}()$ denotes the indicator function and $V_d^{(\varepsilon)}$ is the volume of a $d$-dimensional ball of radius $\varepsilon$. I notice that the normalisation never needs to be computed since the

[1] A kernel function is a continuous non-negative even function that integrates to 1.

threshold is given in terms of a distance and a minimum number of points, rather than the implied density threshold: $\tau = MinPts/NV_d^{(\varepsilon)}$.

Since DBSCAN is based on a distance measure, which is a binary function, I want to replace the distance measure with a new one based on relative density, where pairwise points are more similar if they are at a local dense area. However, the relative density measure will still critically depend on the definition of the neighbourhood, as I discussed above.

In order to avoid using a fixed-size neighbourhood, I use relative mass rather than relative density. I can convert a density-based clustering algorithm to a mass-based clustering algorithm by simply replacing a distance-based density estimator with a tree-based mass estimator [72]. This is shown in the MassTER paper [71] using DBSCAN as well. There is no need to worry about the volume of the neighbourhood and this leads to fewer parameters. Notice that the weakness of finding clusters with varied densities remains for both DBSCAN and MassTER because they are not using a ratio value or relative measure. As a result, they can both overcome this weakness by using a relative measure, i.e., relative density (density-ratio) for density-based algorithms and relative mass for mass-based algorithms. The relative mass description is as follows.

Assuming a method for partitioning the space into regions, I can calculate the probability of a new data point falling into a small region $R(x)$ around $x$ with respect to the probability of the new data point falling into a larger region $R(x,y)$ covering both $x$ and $y$. Notice that I am effectively using a second data point $y$ to define a neighbourhood around $x$. This conditional probability can be estimated as follows

$$\widehat{P}(R(x) \mid R(x,y)) = \frac{N_{R(x)}}{N_{R(x,y)}} \tag{3.9}$$

where $N_{R(x)} = |\{x_i \in D | x_i \in R(x)\}|$ denotes the number of points from the dataset $D$ that lie in the region $R(x)$. Assuming the regions are nested but non-overlapping, i.e., $R(x) \subseteq R(x,y)$, the conditional probability will decrease in a stepwise manner as the distance to $y$ increases. In order to produce a data-dependent kernel function based on

this measure, I need to symmetrise the value by averaging the conditional probabilities of both $R(x)$ and $R(y)$

$$\widetilde{K}_x(y) \propto \mathbf{1}(\frac{P(R(x) \mid R(x,y)) + P(R(y) \mid R(x,y))}{2} \geqslant \frac{1}{\varepsilon}) \tag{3.10}$$

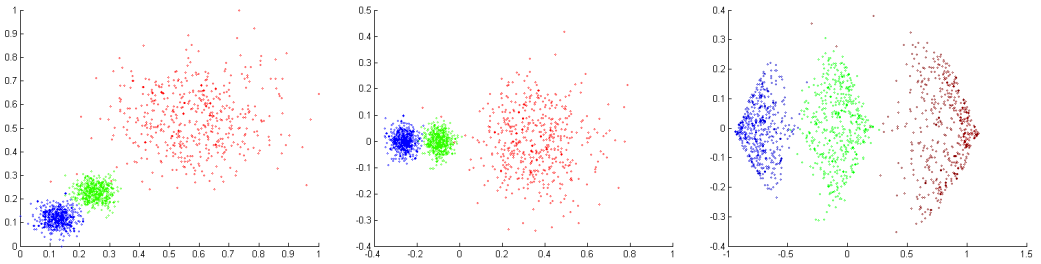
I notice that there is no need to explicitly normalise the kernel (since the parameters $MinPts$ and $\varepsilon$ implicitly define a density threshold) and that $\varepsilon$ appears now as a denominator since closer data points result in higher conditional probability. Intuitively, the revised density estimator $\widehat{pdf}(x)$ (produced by this modified kernel $\widetilde{K}_x(y)$) calculates the percentage of data points $y$ for which the (conditional) probability of points lying in the local regions around $x$ or $y$ is high with respect to the region containing both points. This should result in a value that is correlated with the density-ratio described in Section 3.3.1, i.e., a point with a higher local mass value has a higher $\widehat{pdf}(x)$ value.

I can treat the averaged conditional probability estimate as a dissimilarity measure between two data points by simply inverting the value. (Note the $\widetilde{K}_x(y)$ above is unaffected.) So given a dataset $D$ and partitioning structure $\mathcal{R}$, I define a new dissimilarity measure as given below. I refer to this measure as mass-based since it is based on counts (without any volume calculation) and as a relative mass value due to the use of the count ratio

$$Z(x,y|D,\mathcal{R}) = \frac{2N_{R(x,y)}}{N_{R(x)} + N_{R(y)}} \tag{3.11}$$

Here $R(*)$ denotes the smallest local region covering * among the regions in $\mathcal{R}$ (where local regions can be defined for any given point or pair of points).

Here I provide an example illustrating the difference between the dissimilarity measure based on Euclidean distance and the dissimilarity measure based on relative mass. The two-dimensional non-STS distribution shown in Figure 3.5(a) contains three Gaussians of varied densities, where the density distribution in each dimension has the same characteristic as that shown in Figure 3.3(a). In order to visualise in two dimensions how far apart points are from one another according to each of these two



(a) Three Gaussians (b) MDS using distance (c) MDS using relative mass

Fig. 3.5 (a) A mixture of three Gaussian distributions having a non-STS distribution; (b) Multidimensional scaling on the Euclidean distance dissimilarity matrix; (c) Multidimensional scaling on the relative mass dissimilarity matrix.

dissimilarity measures, I perform multidimensional scaling (MDS)[2]. The MDS plot based on a distance matrix is shown in Figure 3.5(b) and the one based on a relative mass matrix is shown in Figure 3.5(c). Notice that, when using the relative mass dissimilarity, all three clusters are transformed to have almost the same density and the gap between two high-density clusters in the original distribution also becomes wider. As a result, a threshold can easily be found to identify all these clusters, shown in Figure 3.5(c). In contrast, the distribution shown in Figure 3.5(b) has similar densities as in the original three Gaussians, where a threshold cannot be found to identify all three clusters.

I describe a method for defining the regions in $\mathcal{R}$, required to compute the relative mass dissimilarity measure, in Section 3.4.3.

## 3.4 Algorithms and Implementations

This section presents the algorithms of the three solutions to overcome the weakness of density-based clustering algorithms.

[2]Multidimensional scaling is a data analysis technique for visualising the information contained in a dissimilarity matrix [12]. The aim of an MDS algorithm is to place each data point in a $p$-dimensional space (where $p$ is specified by a user), while preserving as much as possible pairwise dissimilarities between them. Notice that the orientation of the reconstruction is arbitrary and that different forms of MDS use different cost criteria for preserving pairwise dissimilarities between points. In this thesis, I use non-metric scaling with Kruskal's non-metric stress criterion for MDS [42] and $p = 2$.

Here I select three state-of-the-art density-based algorithms, DBSCAN, SNN and OPTICS, and apply the proposed three methods to them. DBSCAN is the basic and classic density-based algorithm. SNN and OPTICS are two algorithms used to identify clusters with varied densities.

I use the prefix 'ReScale-' to denote an algorithm which employs the rescaled dataset. 'ReCon-' and 'ReMass-' are applied to the original datasets.

### 3.4.1 Reconditioning DBSCAN, SNN and OPTICS

Based on the analysis given in the last section, the reconditioned DBSCAN algorithm, ReCon-DBSCAN, is shown in Algorithm 1. The algorithm is the same as DBSCAN except: (1) an additional density is estimated using $\eta$-neighbourhood in order to compute density-ratio and (2) a density-ratio threshold is used instead of a density threshold.

Compared with DBSCAN, there is one more parameter $\eta$, which should be set larger than $\varepsilon$. I replace the parameter *MinPts* with $\tau$ to represent the threshold on the density-ratio. Notice that $\tau$ should be set in the range $[0,1]$ because the range of $\frac{|\mathcal{N}_\varepsilon(x)|}{|\mathcal{N}_\eta(x)|}$ is in $[0,1]$.

Because SNN employs DBSCAN as its core, ReCon-SNN also uses ReCon-DBSCAN. The dissimilarity measure used in both SNN and ReCon-SNN is based on the number of shared $k$-nearest neighbours, i.e., $k$ minus the number of shared nearest neighbours.

OPTICS employs a density estimator which is based on *kNN* distance, where $k = MinPts - 1$. A point with short *kNN* distance is likely to be located in a valley in the reachability plot and it is extracted as a cluster centre. ReCon-OPTICS computes a ratio based on two *kNN* distances with two $k$ values, where one is less than the other, i.e., $\alpha < \beta$, to calculate the reachability distance ratio as follows

$$ReachRatio(x_i, x_j, \alpha, \beta) = \frac{max(Core\text{-}dist(x_i, \alpha), \|x_i - x_j\|)}{max(Core\text{-}dist(x_i, \beta), \|x_i - x_j\|)}$$

**Algorithm 1** ReCon-DBSCAN($D$, $\varepsilon$, $\eta$, $\tau$)

**Input:** $D$: input data ($n \times d$ matrix); $\varepsilon$: radius of the neighbourhood; $\eta$: radius of the larger neighbourhood; $\tau$: threshold on density-ratio
**Output:** $C$: Class labels

Mark all points as *unvisited*
**while** exist *unvisited* points in $D$ **do**
    Select an *unvisited* point $x$ and mark it as *visited*
    $\mathcal{N}_\varepsilon(x) \leftarrow \{y \in D \mid \|x-y\| \leqslant \varepsilon\}$
    $\mathcal{N}_\eta(x) \leftarrow \{y \in D \mid \|x-y\| \leqslant \eta\}$
    **if** $\frac{|\mathcal{N}_\varepsilon(x)|}{|\mathcal{N}_\eta(x)|} \geqslant \tau$ **then**
        Create a new cluster $C$, and add $x$ to $C$
        $N \leftarrow \mathcal{N}_\varepsilon(x)$
        **for** each point $x'$ in $N$ **do**
            **if** $x'$ is *unvisited* **then**
                Mark $x'$ as *visited*
                $\mathcal{N}_\varepsilon(x') \leftarrow \{y \in D \mid \|x'-y\| \leqslant \varepsilon\}$
                $\mathcal{N}_\eta(x') \leftarrow \{y \in D \mid \|x'-y\| \leqslant \eta\}$
                **if** $\frac{|\mathcal{N}_\varepsilon(x')|}{|\mathcal{N}_\eta(x')|} \geqslant \tau$ **then**
                    $N \leftarrow N \cup \mathcal{N}_\varepsilon(x')$
                **end if**
            **end if**
            **if** $x'$ is not yet a member of any cluster **then**
                Add $x'$ to $C$
            **end if**
        **end for**
        Output $C$
    **else**
        Mark $x$ as *NOISE*
    **end if**
**end while**

where $x_i \neq x_j \in D$ and $\mathit{Core\text{-}dist}(x_i, \alpha)$ is a function returning the distance between $x_i$ and its $(\alpha-1)^{th}$ nearest neighbour.

ReCon-OPTICS plots the reachability distance ratio instead of the reachability distance according to the same point order as OPTICS with $\mathit{MinPts} = \alpha$. The same procedure is used for both OPTICS and ReCon-OPTICS to extract clusters from this reachability plot.

Since $\eta$ should be larger than $\varepsilon$ for both ReCon-DBSCAN and ReCon-SNN, I re-parameterise $\eta = \lambda\varepsilon$ and then I only need to set $\lambda > 1$ to control the gap between

$\varepsilon$ and $\eta$. For ReCon-OPTICS, I also re-parameterise $\beta = \alpha + \omega$ and use the positive integer $\omega$ to control the gap between $\alpha$ and $\beta$.

### 3.4.2 ReScaling a Dataset

As indicated in Section 3.3.2, ReScaling a dataset $D$ can be implemented in three steps, as shown in Algorithm 2. Let $A_i$ be the attribute value of $D$ projected on the $i^{th}$ dimension. First, for each attribute, it builds a set $E_i$ representing the density distribution in the $\eta$-neighbourhood of the data $A^i$, as shown in Algorithm 3. After that, it scales $A_i$ to yield $S_i$ based on the cumulative distribution of $E_i$, as illustrated in Algorithm 4. Having ReScaled the data, the values for $y$ in each dimension of $S$ will lie between 0 and 1.

I utilise a histogram to estimate the density of $\eta$-neighbourhood in Algorithm 3 because it retains that ReScale has linear time complexity.

**Algorithm 2** ReScale($D$, $\iota$, $\eta$)

**Input:** $D$: input data ($n \times d$ matrix); $\iota$: number of intervals; $\eta$: radius of the neighbourhood
**Output:** $S$: dataset after scaling
1: $d \leftarrow$ dimensionality of $D$
2: **for** $i = 1$ to $d$ **do**
3: $\quad E_i \leftarrow \text{AvgNeighDensity}(A_i, \iota, \eta)$
4: $\quad S_i \leftarrow \text{ScalingOneD}(A_i, E_i, \iota)$
5: **end for**
6: **return** $S$

**Algorithm 3** AvgNeighDensity($A, \iota, \eta$)

**Input:** $A$: 1-dimensional dataset; $\iota$: number of intervals; $\eta$: radius of the neighbourhood
**Output:** $E$: $\{\chi_i, \kappa_i, i = 1, \ldots, \iota + 1\}$ the density distribution of $\eta$-neighbourhood
1: Initialise $E$
2: Split the space into $\iota$ equal intervals with $\iota + 1$ boundary positions $\{\chi_1, \chi_2, ..., \chi_{\iota+1}\}$
3: **for** $i = 1$ to $\iota + 1$ **do**
4: $\quad \kappa_i = \sum_{x \in A} \mathbf{1}(\|\chi_i - x\| \leqslant \eta)$
5: **end for**
6: $E \leftarrow \{\chi, \kappa\}$
7: **return** $E$

**Algorithm 4** ScalingOneD($A, E, \iota$)

**Input:** $A$: 1-dimensional dataset; $E$: the density distribution of $\eta$-neighbourhood; $\iota$: number of intervals
**Output:** $S$: dataset after scaling

```
1: n = |A|
2: Initialise S = {y_i = 0, i = 1, ..., n}
3: for i = 1 to n do
4:     for j = 1 to ι + 1 do
5:         if x_i ⩾ χ_j then
6:             y_i = y_i + κ_j
7:         end if
8:     end for
9: end for
10: Normalise S to be in the range [0, 1]
11: return S
```

### 3.4.3 Relative Mass Dissimilarity

Recursive partitioning-based measures have been used successfully in the past to perform clustering and anomaly detection [71] [72]. They are applicable to this problem because we can calculate the relative measure based on partitioning.

I use a recursive partitioning scheme called *iForest* [6, 47, 49] to define regions. *iForest* uses a collection of randomly split trees to detect global anomalies. A similar scheme is used to define mass estimation [72]. *iForest* has been improved to find local anomalies with a unary function of relative mass [6]. In this thesis, I create a new dissimilarity measure based on relative mass which is a binary function.

The implementation can be divided into two steps. There are two input parameters to this procedure, namely, the number of trees $t$ and the sub-sampling size $\psi$. The height limit $h$ is automatically set by $\psi : h = \lceil log_2 \psi \rceil$.

The first step is to build an *iForest* consisting of $t$ *iTrees* as the partitioning structure $\mathcal{R}$. Each *iTree* is built independently using a subset $\mathcal{D}$ sampled without replacement from $D$, where $|\mathcal{D}| = \psi$. A random splitting mechanism is used at each *iTree* node to partition the current sample set into two non-empty subsets until every point is isolated or the maximum tree height $h$ is reached. The details of the *iTree* building process can be found in Algorithm 9 in Appendix A.

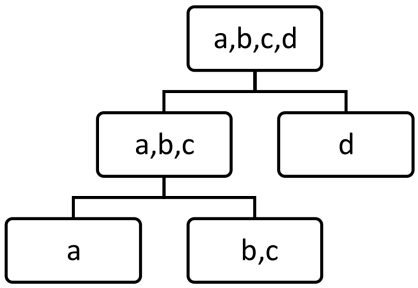



Fig. 3.6 Example: four points partitioned by a 2-level *iTree*.

After an *iForest* has been built, all points in $D$ are traversed through each tree in order to record the mass (the number of data points) of each node for evaluation.

The second step is the evaluation step. Test points $x$ and $y$ are passed through each *iTree* to calculate the ratio of the sum of the mass of the lowest nodes containing $x$ and $y$ independently to the mass of the lowest node containing both $x$ and $y$. Finally, $\overline{Z}(x,y)$ is the harmonic mean of these ratios as defined below

$$\overline{Z}(x,y) \approx (\frac{1}{t}\sum_{k=1}^{t} Z(x,y|\mathcal{R}_k)^{-1})^{-1} \tag{3.12}$$

I notice that the harmonic mean is appropriate here since I am averaging inverted conditional probability estimates, as discussed in the previous section.

Figure 3.6 illustrates an example of four points partitioned by an *iTree*. Based on this *iTree*, $Z(a,b) = (2\times 3)/(1+2) = 2$ and $Z(a,d) = (2\times 4)/(1+1) = 4$. Thus, point $a$ is less similar to point $d$ than to point $b$ because $Z(a,b) < Z(a,d)$.

The value of $Z(x,y)$ is between 1 and $n = |D|$: it is 1 when both $x$ and $y$ are in the same node; it is $n$ when $x$ and $y$ are at different nodes with the mass value of 1 and only at the root node when both $x$ and $y$ are at the same node.

I can now simply replace the distance measure in an existing density-based clustering algorithm with the new measure, in order to create its ReMass version.

### 3.4.4 A Comparison of the Three Approaches

The characteristics of the three approaches are given as follows:

• **Algorithm-specific vs generic**. The ReCon approach is algorithm-specific and each density-based algorithm must be modified to incorporate the density-ratio. The ReScale and ReMass approaches are generic and can be applied to any existing density-based algorithm without modification. The difference between the ReScale and ReMass approaches is that the latter can only be applied to an algorithm relying on a distance (or dissimilarity) calculation.

• **Parameters**. The ReCon approach needs one additional parameter ($\eta$) to those employed in DBSCAN. The ReScale approach and ReMass approach need two additional parameters ($\eta$ and $\iota$ for ReScale, $t$ and $\psi$ for ReMass), Although they can usually be set to default values.

• **Computational complexity**. All three approaches retain the same time and space complexities of the original clustering algorithms. The ReCon approach requires the density estimator used by the density-based algorithm to make additional density estimation with a larger radius in order to compute the density-ratio. Both ReScale and ReMass are a pre-processing step that must be applied before a density-based clustering algorithm can be executed. Table 3.2 gives the time and space complexities of the different algorithms used in the empirical evaluation in the next section. Notice that $\psi$ in ReMass must be less than (or linear in) the data size $n$, so that $O(n+\psi) = O(n)$. The same is true for the parameter $\iota$ in ReScale.

Table 3.2 DBSCAN, OPTICS, SNN and their transformed versions (ReCon, ReScale or ReMass) have the same time and space complexities.

| Algorithm | Time complexity | Space complexity |
|---|---|---|
| ReScale only | $O(dn\iota)$ | $O(dn+d\iota)$ |
| ReMass only | $O(t\psi log\psi + tnlog\psi)$ | $O(tlog\psi + n)$ |
| DBSCAN or its transformed version | $O(dn^2)$ | $O(dn)$ |
| OPTICS or its transformed version | $O(dn^2)$ | $O(dn)$ |
| SNN or its transformed version | $O(k^2n^2 + dn^2)$ | $O(dn+kn)$ |

## 3.5 Empirical Evaluation

This section presents the experiments designed to evaluate the ReCon, ReScale and ReMass approaches. The aim is to assess their effectiveness in overcoming the

weaknesses of existing density-based clustering algorithms in identifying all clusters with varied densities.

### 3.5.1 Experimental Methodology

For each algorithm, I build a dissimilarity matrix in order to avoid distance recomputations for neighbour searching. Because *iForest* is a randomised method, I report the average result of ReMass approach over 10 trials. Because other approaches are deterministic, I only report their results over 1 trial.

#### Datasets

I use 2 synthetic datasets, S1 and S2, and 18 real-world datasets from the UCI Machine Learning Repository [46] to ascertain the abilities of the proposed three approaches in handling datasets with varied densities. Table 3.3 presents the properties of the datasets, which have different data sizes and dimensions. Please see Appendix B for detailed descriptions of these real-world datasets.

The S1 and S2 datasets are used because of their contrasting data characteristics, which I am examining in this investigation. S1 is a particular non-STS distribution which contains three Gaussian clusters $N(\mu,\sigma)$ with means located at $(x^{(1)},x^{(2)})$=(5, 8), (8, 5), (12, 11) and standard deviations $\sigma$=2, 2, 4 in the two dimensions; and each cluster has 300 points. S2 is an example of STS distribution which has three Gaussian clusters $N(\mu,\sigma)$ with means located at $(x^{(1)},x^{(2)}) = (10,10),(20,20),(60,60)$ and $\sigma = 2,2,11$; and each cluster has 500 points. The density plots of S1 and S2 are shown in Figure 3.7.

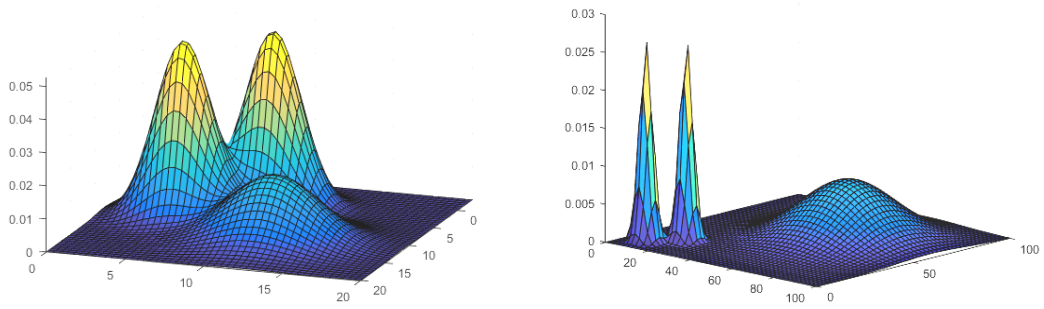
All datasets are normalised using the *min-max* normalisation to yield each attribute value in the range [0,1] before the experiments begin.

#### Evaluation Measures Used

The clustering performance is measured in terms of the F-measure, as discussed in Section 2.3.2. Given a clustering result, I calculate the precision and recall values for each cluster based on the contingency matrix and then calculate the F-measure. I

Table 3.3 Data properties, sorted by data size.

| Dataset | $n$ (#Points) | $d$ (#Dimensions) | $\varsigma$ (#Clusters) |
|---|---|---|---|
| Iris | 150 | 4 | 3 |
| Wine | 178 | 13 | 3 |
| WPBC | 194 | 33 | 2 |
| Sonar | 208 | 60 | 2 |
| Seeds | 210 | 7 | 3 |
| Glass | 214 | 9 | 6 |
| Thyroid | 215 | 5 | 3 |
| Liver | 345 | 6 | 2 |
| Ionosphere | 351 | 33 | 2 |
| Dermatology | 358 | 34 | 6 |
| Musk | 476 | 166 | 2 |
| WDBC | 569 | 30 | 2 |
| ILPD | 579 | 9 | 2 |
| Breast | 699 | 9 | 2 |
| PIMA | 768 | 8 | 2 |
| S1 | 900 | 2 | 3 |
| Hill | 1212 | 100 | 2 |
| S2 | 1500 | 2 | 3 |
| Segment | 2310 | 19 | 7 |
| Spam | 4601 | 57 | 2 |




(a) S1: non-STS distribution (b) S2: STS distribution

Fig. 3.7 The density distributions of the S1 and S2 datasets.

report the best clustering performance within a reasonable range of the parameters searched for each algorithm. Table 3.4 lists the parameters and their search range for each algorithm. Notice that the performance of SNN is highly sensitive to the value of $k$ [21], [59]. In this thesis, $k$ in SNN is set to the square root of the data size as suggested by previous research [23], [79]. The parameter $\xi$ in OPTICS is used to

identify downward and upward areas of the reachability plot in order to extract all clusters using a hierarchical method [5].

Table 3.4 Parameters and their search ranges for each algorithms.

| Algorithm | Parameter with search range |
|---|---|
| DBSCAN | $MinPts \in \{2,3,...,10\}$; $\varepsilon \in [0,1]$ |
| ReCon-DBSCAN | $\tau \in [0,1]$; $\varepsilon \in [0,1]$; $\lambda \in \{1.1,1.2,...,2.0\}$ |
| SNN | $MinPts \in \{2,3,...,10\}$; $k = \sqrt{\lvert D \rvert}$; $\varepsilon \in [1,k]$ |
| ReCon-SNN | $\tau \in [0,1]$; $k = \sqrt{\lvert D \rvert}$; $\varepsilon \in [1,k]$; $\lambda \in \{1.1,1.2,...,2.0\}$ |
| OPTICS | $MinPts \in \{2,3,...,10\}$; $\xi \in \{0.01,0.02,...,0.99\}$ |
| ReCon-OPTICS | $\alpha \in \{2,3,...,10\}$; $\omega \in \{2,3,...,10\}$; $\xi \in \{0.01,0.02,\ldots,0.99\}$ |
| ReScale | $\iota \in \{10,100,1000\}$; $\eta \in \{0.1,0.2,...,0.5\}$ |
| ReMass | $\psi \in \{2,4,8...,256\}$; $t \in \{10,100,1000\}$ |

#### Implementation Platform

All algorithms used in my experiments are implemented in Matlab. The experiments are run on a machine with four cores (Intel Core i7-3770 3.40GHz) and 16GB memory.

### 3.5.2 Clustering Performance

In this section, I first analyse the performance of different algorithms divided into three groups according to their basic algorithms (DBSCA, OPTICS and SNN). Then I compare the top performers from each group and provide my recommendation for real applications.

#### DBSCAN and its Transformed Versions

Table 3.5 shows the best F-measure of DBSCAN and its ReCon, ReScale and ReMass versions. The win/loss counts in the last column of the table reveal that ReScale-DBSCAN and ReMass-DBSCAN performed better than DBSCAN in all 20 datasets.

In terms of the average F-measure, ReMass-DBSCAN has the highest average of 0.67. ReScale-DBSCAN and ReCon-DBSCAN have the average F-measure 0.65 and 0.64, respectively.

Table 3.5 Best F-measure of DBSCAN and its transformed clustering algorithms on the 20 datasets. The best performer on each dataset is in boldface.

| Algorithm | DBSCAN | ReCon-DBSCAN | ReScale-DBSCAN | ReMass-DBSCAN |
|---|---|---|---|---|
| Iris | 0.87 | **0.95** | 0.91 | 0.93 |
| Wine | 0.65 | **0.90** | 0.88 | 0.90 |
| WPBC | 0.45 | **0.51** | 0.49 | 0.51 |
| Sonar | 0.38 | **0.46** | 0.41 | 0.43 |
| Seeds | 0.75 | 0.89 | **0.91** | 0.89 |
| Glass | 0.37 | **0.52** | 0.51 | 0.49 |
| Thyroid | 0.58 | 0.75 | 0.79 | **0.90** |
| Liver | 0.37 | 0.43 | **0.56** | 0.45 |
| Ionosphere | 0.50 | 0.58 | 0.51 | **0.61** |
| Dermatology | 0.51 | 0.69 | **0.73** | 0.62 |
| Musk | 0.36 | **0.54** | 0.52 | 0.52 |
| WDBC | 0.60 | **0.86** | 0.80 | 0.86 |
| ILPD | 0.40 | 0.46 | 0.44 | **0.48** |
| Breast | 0.49 | 0.49 | **0.95** | 0.95 |
| PIMA | 0.43 | **0.57** | 0.56 | 0.50 |
| S1 | 0.34 | **0.74** | 0.70 | 0.68 |
| Hill | 0.29 | 0.29 | 0.34 | **0.46** |
| S2 | 0.92 | 0.97 | **0.99** | 0.99 |
| Segment | 0.59 | **0.71** | 0.62 | 0.66 |
| Spam | 0.38 | 0.42 | 0.42 | **0.49** |
| Average | 0.51 | 0.64 | 0.65 | **0.67** |
| Win/Loss | – | 18/0 | 20/0 | 20/0 |

Notably in the S1 dataset having non-STS distribution, all three transformed versions achieved more than double the F-measure of DBSCAN. Even on the S2 dataset having STS distribution, all three transformed versions also outperformed DBSCAN because more points were assigned to the correct clusters.

All three approaches improve by 20% on the performance of DBSCAN on three datasets: Wine, WDBC and S1 datasets. In addition, ReScale-DBSCAN improves by 20% on the performance of DBSCAN on three more datasets: Thyroid, Dermatology and Breast datasets.

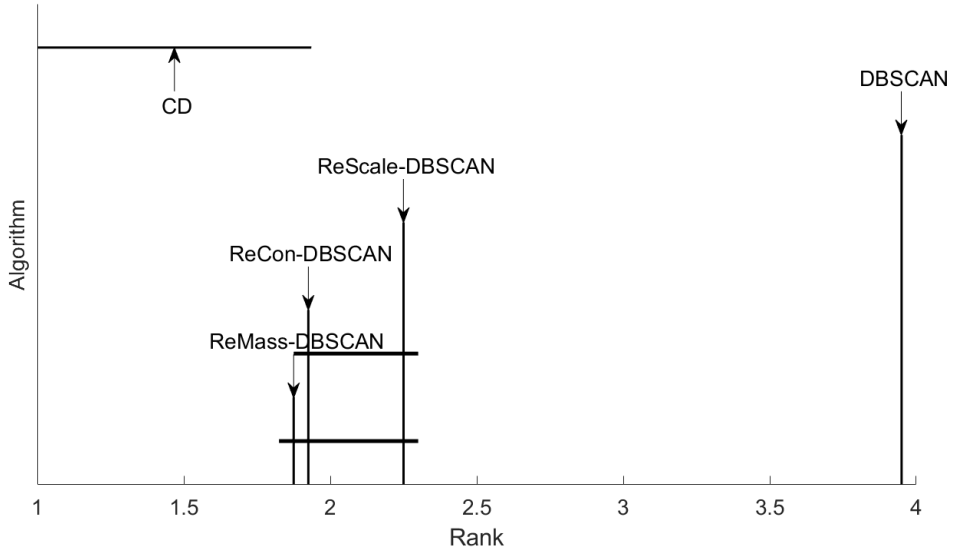



Fig. 3.8 Critical difference (CD) diagram of the post-hoc Nemenyi test ($\alpha = 0.1$). The difference between two algorithms is significant if the gap between their ranks is larger than the CD. There is a line between two algorithms if the rank gap between them is smaller than the CD.

I conduct a Friedman test with the post-hoc Nemenyi test [19] to examine whether the difference between any two algorithms is significant. First, I rank the algorithms on each dataset according to their average F-measure, where the best is rank 1. Then I use the post-hoc Nemenyi test to calculate the critical difference (CD) value for each algorithm. Figure 3.8 shows each algorithm's average rank and CD. This test also shows that ReMass-DBSCAN is the best performer, and all transformed DBSCAN versions are significantly better than the original DBSCAN.

### OPTICS and its Transformed Versions

Table 3.6 shows the best F-measure of OPTICS and its ReCon, ReScale and ReMass versions. ReScale-OPTICS has the highest average F-measure of 0.77, followed by ReMass-OPTICS with F-measure of 0.75. The win/loss counts in the last column of the table reveal that ReScale-OPTICS performed better than OPTICS on 15 datasets.

Figure 3.9 shows the Friedman test with the post-hoc Nemenyi test [19] to examine whether the difference between any two algorithms is significant. It can be seen that ReScale-OPTICS is the best performer and significantly better than the original OPTICS.

Table 3.6 Best F-measure of OPTICS and its transformed clustering algorithms on the 20 datasets. The best performer on each dataset is in boldface.

| Algorithm | OPTICS | ReCon-OPTICS | ReScale-OPTICS | ReMass-OPTICS |
|---|---|---|---|---|
| Iris | 0.85 | **0.91** | 0.90 | 0.88 |
| Wine | 0.76 | 0.80 | 0.84 | **0.86** |
| WPBC | **0.65** | 0.59 | 0.65 | 0.63 |
| Sonar | 0.67 | 0.49 | **0.67** | 0.67 |
| Seeds | 0.80 | 0.81 | **0.89** | 0.89 |
| Glass | 0.55 | 0.51 | **0.65** | 0.56 |
| Thyroid | 0.59 | 0.59 | 0.85 | **0.86** |
| Liver | 0.66 | 0.57 | **0.67** | 0.66 |
| Ionosphere | 0.77 | 0.49 | **0.85** | 0.73 |
| Dermatology | 0.75 | 0.83 | **0.88** | 0.75 |
| Musk | 0.67 | 0.58 | **0.68** | 0.67 |
| WDBC | 0.68 | 0.56 | 0.76 | **0.83** |
| ILPD | 0.68 | 0.57 | **0.71** | 0.70 |
| Breast | 0.84 | 0.46 | **0.96** | 0.94 |
| PIMA | 0.65 | 0.65 | **0.66** | 0.65 |
| S1 | 0.53 | 0.54 | **0.70** | 0.70 |
| Hill | **0.67** | 0.60 | 0.66 | 0.62 |
| S2 | 0.98 | 0.98 | **0.99** | 0.99 |
| Segment | 0.70 | 0.67 | 0.67 | **0.72** |
| Spam | **0.69** | 0.66 | 0.66 | 0.66 |
| Average | 0.71 | 0.64 | **0.77** | 0.75 |
| Win/Loss | – | 5/12 | 15/3 | 11/4 |

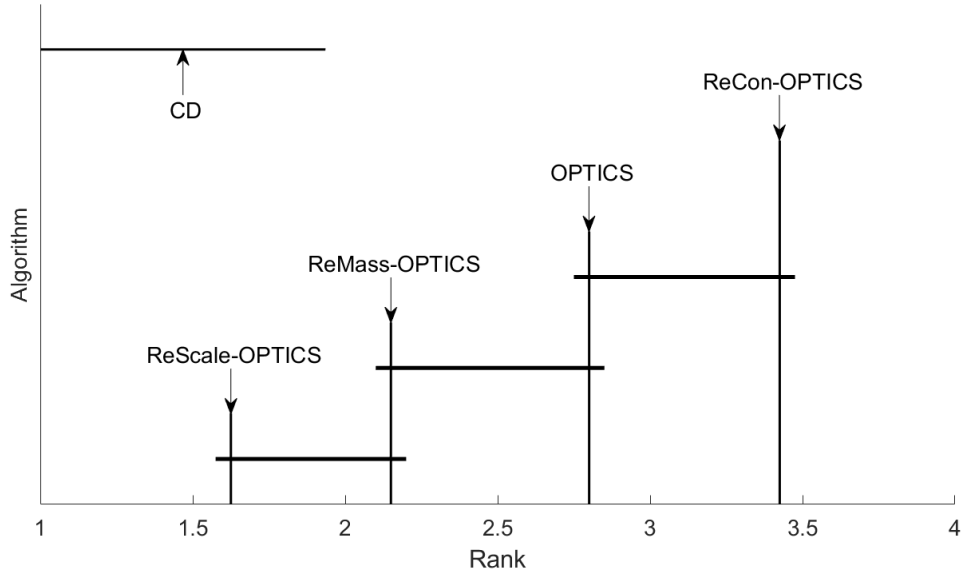
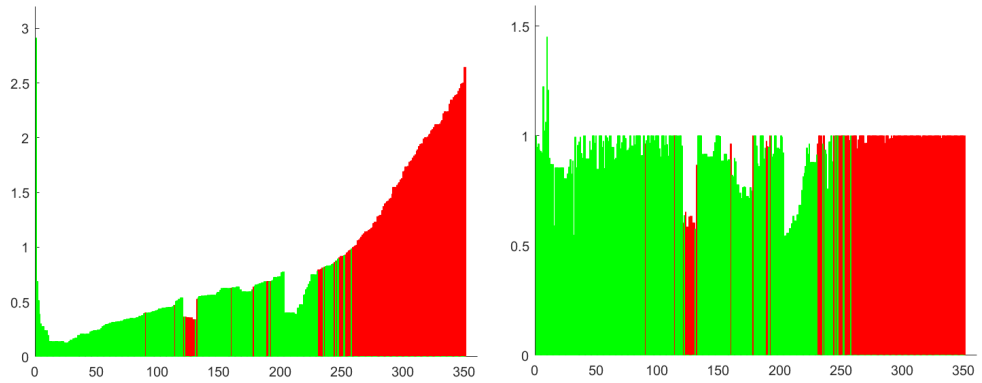



Fig. 3.9 Critical difference (CD) diagram of the post-hoc Nemenyi test ($\alpha = 0.1$). The difference between two algorithms is significant if the gap between their ranks is larger than the CD.




(a) Reachability plot of OPTICS (b) Reachability plot of ReCon-OPTICS

Fig. 3.10 Reachability plot of OPTICS ($MinPts = 10$) and ReCon-OPTICS($\alpha = 10, \omega = 5$) on Ionosphere dataset. Each colour indicates a true cluster.

It is interesting to see that the ReCon approach only improves the performance of OPTICS on 5 datasets while decreasing the performance of OPTICS on 12 datasets. The reason is that the reachability ordering may not strictly separate the clusters, i.e., a valley in the plot may contain points from different clusters. The ReCon approach does not change the ordering, but makes the reachability plot more smooth or flat. Therefore, a hierarchical method will be less able to group more points around each valley in the plot. An example is shown in Figure 3.10.

In contrast, ReScale and ReMass can significantly improve the ordering such that the points from the same cluster are close to each other. Then, clusters are more pure and easier to be identified. I will discuss this aspect further in Section 3.5.3.

### SNN and its Transformed Versions

Table 3.7 shows the best F-measure of SNN and its ReCon, ReScale and ReMass versions. ReScale-SNN has the highest average F-measure of 0.69. Both ReCon-SNN and ReMass-SNN have F-measure of 0.68. ReScale-SNN performed better than OPTICS on 17 datasets.

Figure 3.11 shows the Friedman test with the post-hoc Nemenyi test [19] for examining whether the difference between any two algorithms is significant. All three approaches improve on the performance of SNN. Both ReScale-SNN and ReMass-SNN are significantly better than original SNN.

Table 3.7 Best F-measure of SNN and its transformed clustering algorithms on the 20 datasets. The best performer on each dataset is in boldface.

| Algorithm | SNN | ReCon-SNN | ReScale-SNN | ReMass-SNN |
|---|---|---|---|---|
| Iris | 0.96 | 0.92 | **0.97** | 0.89 |
| Wine | 0.91 | 0.91 | 0.95 | **0.96** |
| WPBC | 0.47 | 0.48 | **0.58** | 0.54 |
| Sonar | 0.45 | 0.44 | **0.52** | 0.47 |
| Seeds | 0.89 | 0.87 | **0.91** | 0.91 |
| Glass | 0.41 | 0.49 | **0.52** | 0.49 |
| Thyroid | 0.85 | 0.84 | **0.88** | 0.87 |
| Liver | 0.45 | 0.52 | **0.56** | 0.45 |
| Ionosphere | 0.50 | 0.50 | 0.58 | **0.68** |
| Dermatology | 0.86 | 0.92 | **0.94** | 0.90 |
| Musk | 0.49 | 0.49 | **0.58** | 0.51 |
| WDBC | 0.70 | 0.84 | **0.90** | 0.78 |
| ILPD | 0.43 | 0.55 | 0.53 | **0.58** |
| Breast | **0.90** | 0.88 | 0.82 | 0.74 |
| PIMA | 0.53 | **0.58** | 0.57 | 0.49 |
| S1 | 0.50 | 0.61 | 0.58 | **0.64** |
| Hill | 0.46 | **0.50** | 0.34 | 0.46 |
| S2 | 0.99 | 0.99 | **0.99** | 0.99 |
| Segment | 0.66 | **0.71** | 0.70 | 0.70 |
| Spam | 0.43 | 0.47 | 0.47 | **0.67** |
| Average | 0.64 | 0.68 | **0.69** | 0.68 |
| Win/Loss | – | 11/5 | 17/2 | 14/3 |

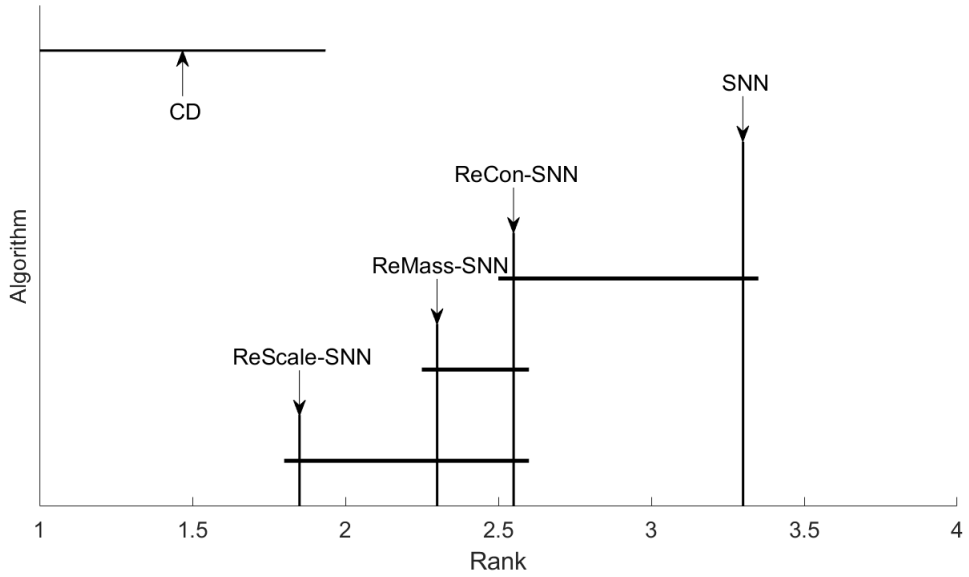



Fig. 3.11 Critical difference (CD) diagram of the post-hoc Nemenyi test ($\alpha = 0.1$). The difference between two algorithms is significant if the gap between their ranks is larger than the CD.

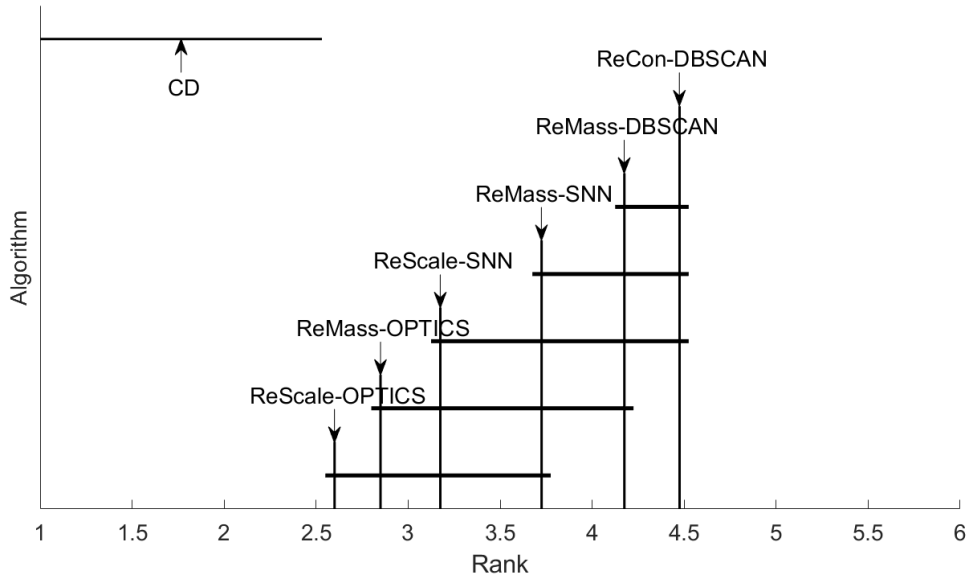



Fig. 3.12 Critical difference (CD) diagram of the post-hoc Nemenyi test ($\alpha = 0.1$). The difference between two algorithms is significant if the gap between their ranks is larger than the CD.

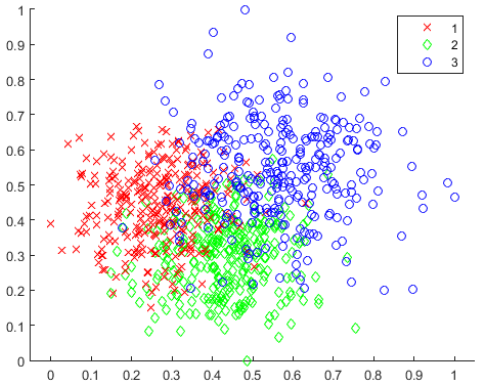
The reason that ReCon does not significantly improve over SNN is because of the fundamental issues with SNN. SNN is a $k$-nearest neighbour-based algorithm, which can be highly sensitive to the setting of the parameter $k$. Thus, ReCon-SNN cannot significantly improve on SNN's performance when $k$ is set to its default value.

It is interesting to see the final comparison of different density-based clustering algorithms. Here I select the top two performers from each group and employ the Friedman test with the post-hoc Nemenyi test [19], as shown in Figure 3.12. Both ReScale-OPTICS and ReMass-OPTICS are also significantly better than ReCon-DBSCAN. ReScale-OPTICS is significantly better than ReMass-DBSCAN.

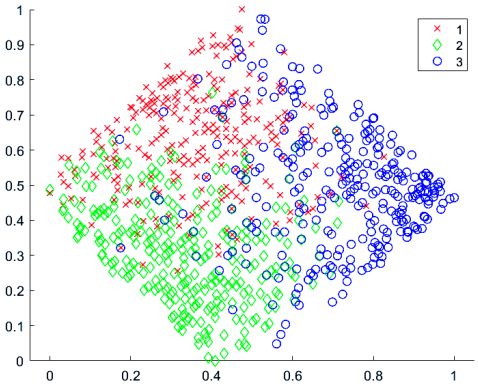
Although I have shown that the ReMass and ReScale versions of OPTICS and SNN perform better than the original algorithms, I still recommend ReMass-DBSCAN for real applications. The reason is that SNN and the hierarchical method used for OPTICS require additional parameters and/or have higher time complexity, but ReMass-DBSCAN retains the same number of parameters and same time complexity as DBSCAN.

### 3.5.3 Visualising the Effects of ReScale and ReMass

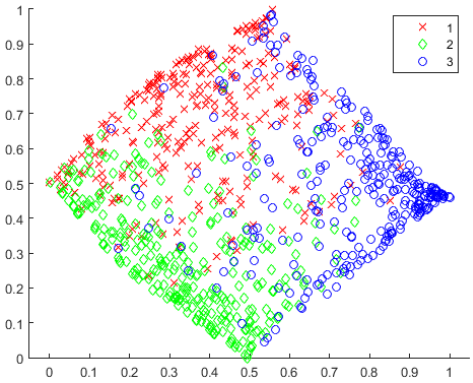
Both ReScale and ReMass aim to transform the original dataset to one which is more uniform. I use multidimensional scaling (MDS) as a convenient means to verify whether ReScale and ReMass have achieved this aim. Here I present the original and the transformed S1 datasets in Figure 3.13.




(a) MDS on original data




(b) MDS on ReScaled data

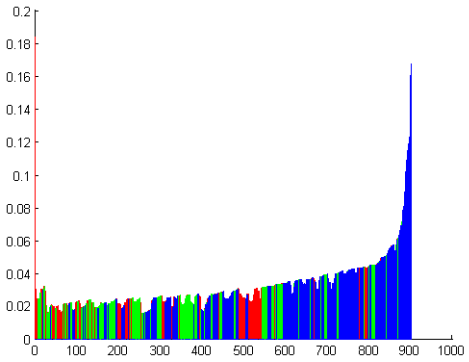



(c) MDS using relative mass

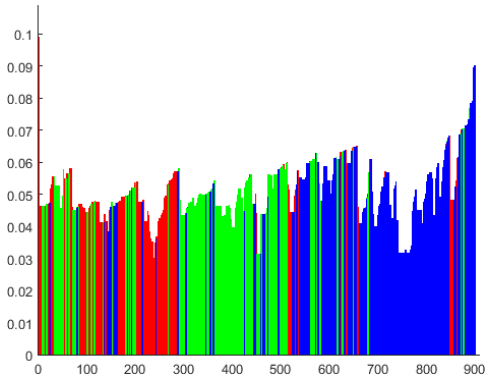
Fig. 3.13 MDS plots in two dimensions on the original and the transformed S1 datasets. Each colour indicates a true cluster.

In the MDS plots, different cluster modes in both the ReScaled and ReMassed datasets are more distanced from each other than those in the original dataset, and the overlapping region between clusters becomes sparser.

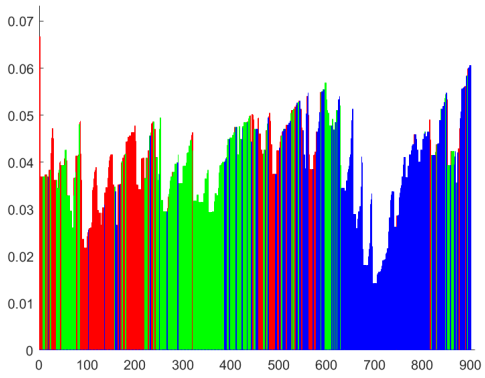
OPTICS [5] attempts to overcome the weakness of DBSCAN by first producing an ordering of the data points with respect to its density-based clustering structure and then extracting clusters based on this order. It assumes that each cluster has its own




(a) Reachability plot on original data




(b) Reachability plot on ReScaled data




(c) Reachability plot using relative mass

Fig. 3.14 Reachability plot of OPTICS ($MinPts = 10$) on the original and the transformed S1 datasets. Because $Z(x,y)$ is between 1 and $n = |D|$, I use $Z(x,y) - 1$ for the reachability plot of ReMass-OPTICS. Each colour indicates a true cluster.

typical reachability scale, and the reachability distance of the cluster boundary should be larger than that of its cluster centre. Hence each cluster shows up as a distinct valley in the reachability plot. However, I have found that a valley of the reachability plot may contain data points from different clusters if some clusters are very close or their boundaries overlap each other. This is because the ordering only follows the nearest neighbour in terms of the reachability distance. Thus, reducing the cluster overlap or enlarging the distance gap between different clusters' boundaries can improve the performance of OPTICS.

Both ReScale and ReMass mould clusters to be more separable from each other and this improves the ordering quality of the reachability plot. Figure 3.14 presents the reachability plots of OPTICS before and after rescaling on the S1 dataset. When comparing the reachability plots in Figure 3.14, the valleys of ReScale-OPTICS

(OPTICS on the ReScaled data) and ReMass-OPTICS (OPTICS using relative mass) are more homogeneous than those of OPTICS on the original S1 dataset. Therefore, clusters are easier to extract using a hierarchical method. Since the valleys are more observable, a reachability plot of the rescaled data is more meaningful for an interactive analysis. The visualisation of ReScale and ReMass on all datasets is in Appendix C. Notice that the visualisation of ReCon is not provided because the ReCon approach does not change the dissimilarity between points or the data distribution.

It is worth mentioning that there are several datasets with significantly overlapping classes observed on their MDS plots: the Glass, Solar, Hill and Liver datasets. All three approaches only slightly improve the performance of original density-based algorithms on these datasets.

### 3.5.4 Parameter Sensitivity

In this section, three investigations are conducted to examine the sensitivity of the new parameters introduced by ReCon, ReScale and ReMass, respectively.

#### ReCon

ReCon only introduces one parameter $\eta$, which is the most important parameter for the density-ratio estimation. I first evaluate the sensitivity of parameter $\eta$ for ReCon. Notice that $\eta$ is re-parameterised as $\lambda$ for ReCon-DBSCAN and ReCon-SNN, and $\beta$ is re-parameterised as $\omega$ for ReCon-OPTICS. Figure 3.15 shows the average F-measure on the 20 datasets when $\lambda$ is from 1.1 to 2.0. The performance of ReCon-DBSCAN decreases as $\lambda$ increases, and it degrades to the performance level of DBSCAN when using $\lambda = 2$; but ReCon-SNN performs the best when using $\lambda = 1.3$. Thus, $\eta$ only needs to be a small value or slightly larger than $\varepsilon$. Recall that a large $\eta$ value will make the density-ratio estimation approximate the true density, and so the ReCon approach will show no advantage.

For ReCon-OPTICS, I investigate $\omega$ in the range $\{2,3,\ldots,10\}$. Figure 3.16 shows the average F-measure on the 20 datasets. The performance of ReCon-OPTICS is similar to the one shown in Table 3.6, i.e., the best setting for $\beta$ is between 10 and 20

more than $\alpha$. The performance of ReCon-OPTICS approximates that of OPTICS with a higher $\omega$ value.

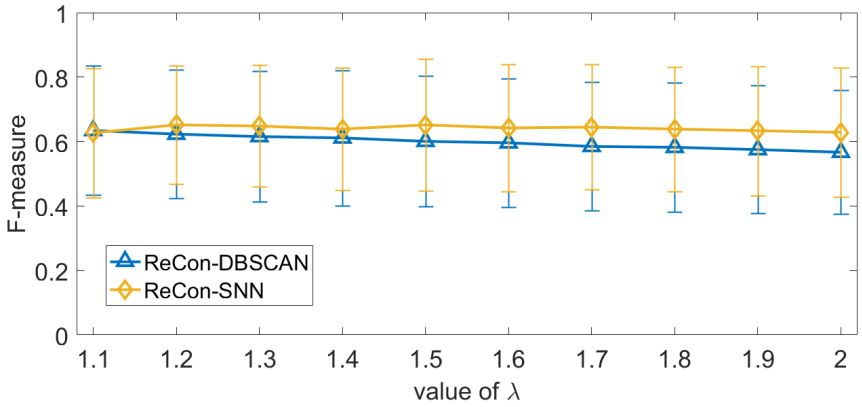



Fig. 3.15 Average F-measure with the standard deviation on 20 datasets.

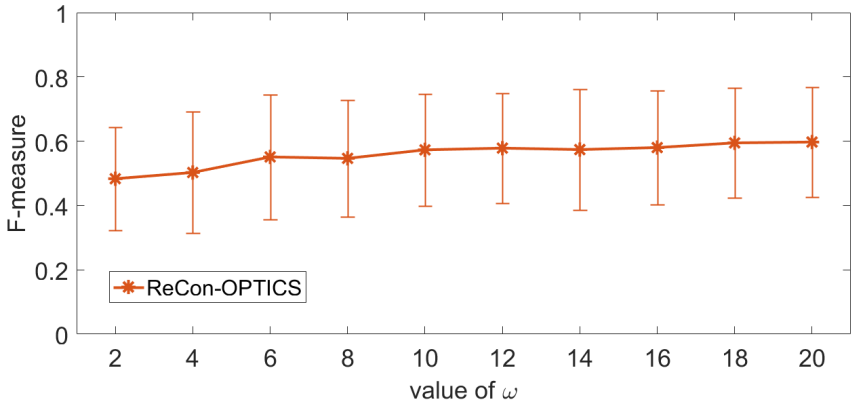



Fig. 3.16 Average F-measure with the standard deviation on 20 datasets.

### ReScale

ReScale has two parameters $\eta$ and $\iota$. I first evaluate the parameter $\eta$ for ReScale. Figure 3.17 shows the average F-measure on 20 datasets when ReScale uses $\eta$ from 0.1 to 0.5 and $\iota = 1000$. The result shows that both ReScale-DBSCAN and ReScale-OPTICS performed well when using $\eta = 0.1$. ReScale-SNN is comparatively less sensitive than ReScale-DBSCAN with different $\eta$ values.

I test different values of $\iota$ for ReScale when $\eta = 0.1$ and found that larger $\iota$ values produce higher F-measure, as shown in Figure 3.18. However, the increase in F-measure when using large $\iota$ values is not significant when $\iota$ is larger than 100.

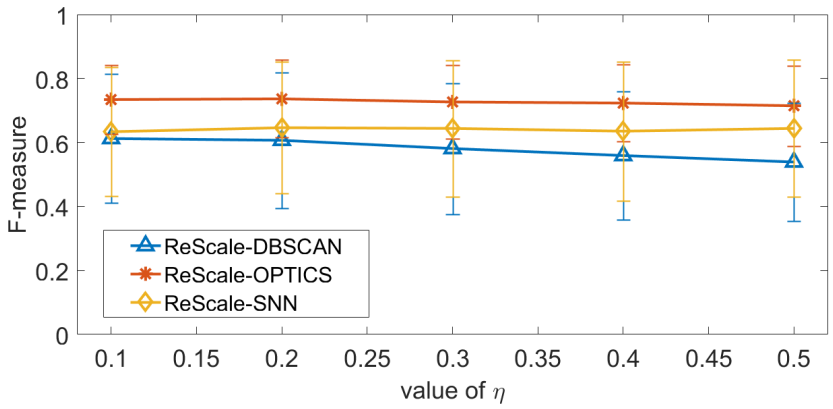



Fig. 3.17 Average F-measure with the standard deviation on 20 datasets when $\iota = 1000$.

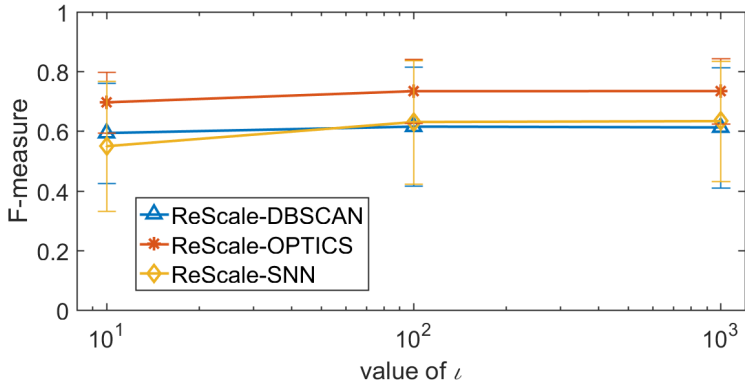
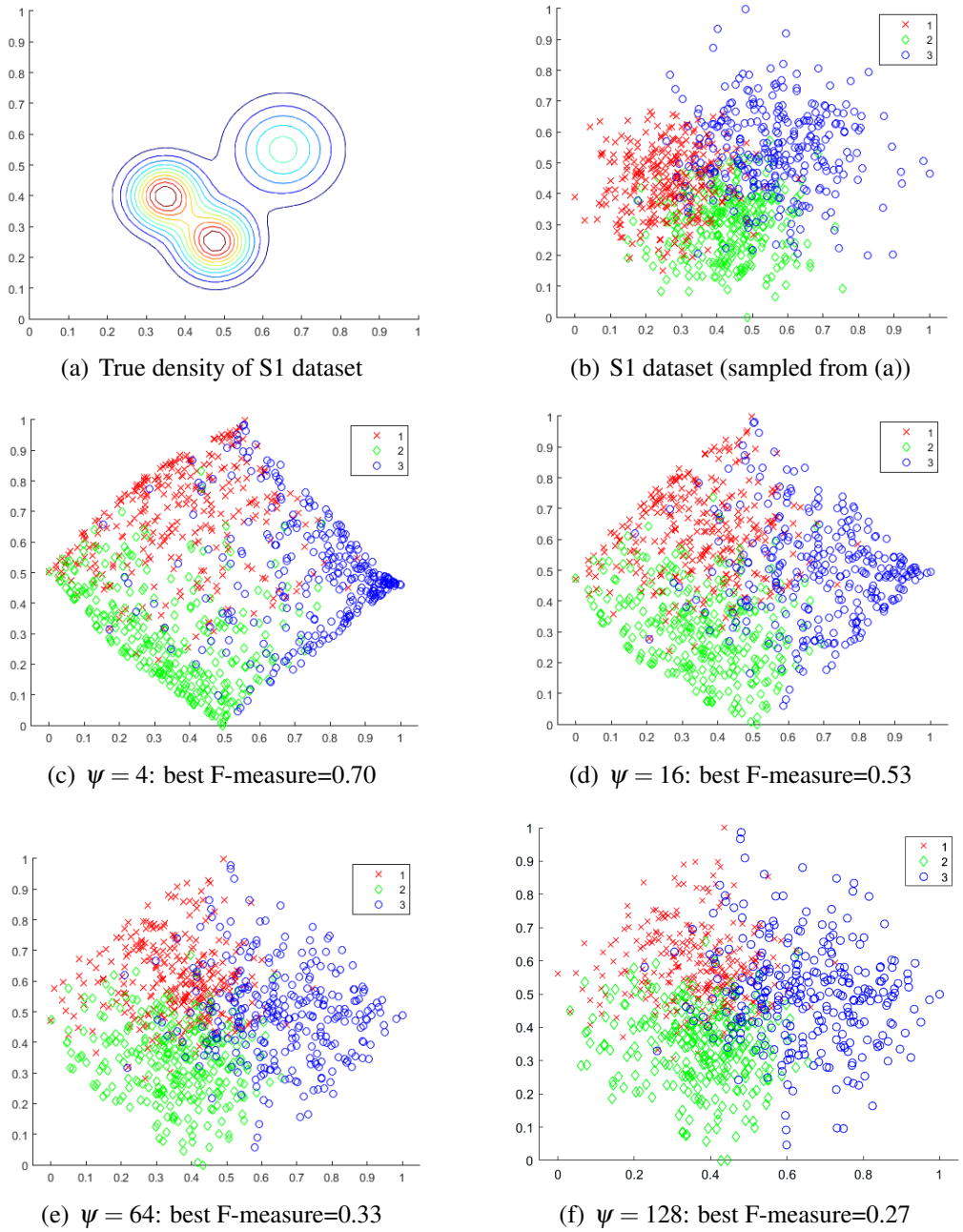



Fig. 3.18 Average F-measure with the standard deviation on 20 datasets when $\eta = 0.1$.

### ReMass

In this part, I evaluate parameters $\psi$ and $t$ for building the relative mass dissimilarity. In order to evaluate the effect of $\psi$, I use the example non-STS distribution S1 dataset, which contains three Gaussians as shown in Figure 3.19(b). Figure 3.19(a) is the contour plot for the original distribution of the S1 dataset. Figure 3.19(c), 3.19(d) and 3.19(e) illustrate the MDS plots using relative mass when $\psi = 4, 16, 64$ and 256, respectively. These plots show that the three Gaussians are more separable using $\psi = 4$ since the majority of the points are at different edges for the three classes and less in the overlapping areas, but their overlapping areas become denser as $\psi$ increases to a high value. This is because large $\psi$ values produce large trees. As a result, $R(x)$ and $R(y)$ are now significantly smaller than $R(x,y)$, yielding an undesirable effect: the relative mass dissimilarity score approximates the distance, and then the estimated density using relative mass approximates the estimated density using distance. Notice

that DBSCAN produced a clustering result which has an equivalent F-measure as that produced by ReMass-DBSCAN using $\psi = 64$.



(a) True density of S1 dataset

(b) S1 dataset (sampled from (a))

(c) $\psi = 4$: best F-measure=0.70

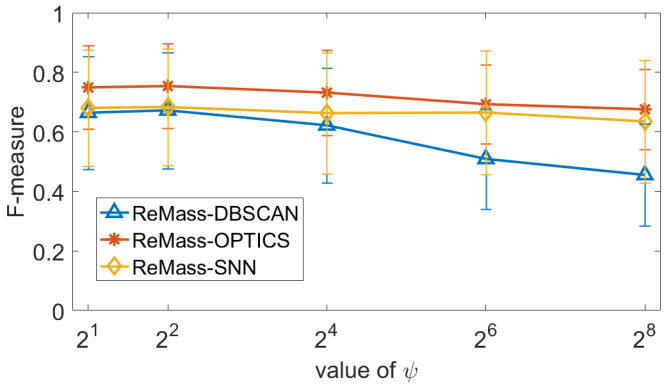
(d) $\psi = 16$: best F-measure=0.53

(e) $\psi = 64$: best F-measure=0.33

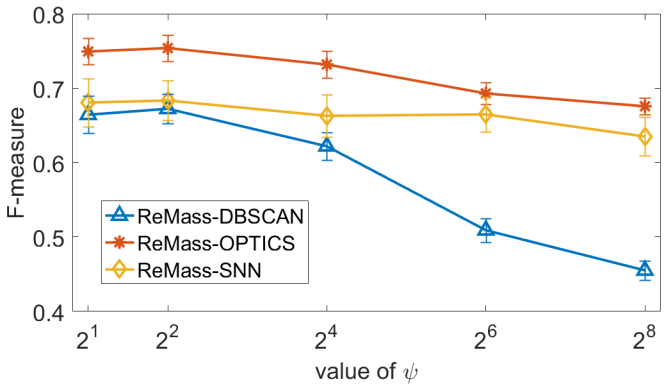
(f) $\psi = 128$: best F-measure=0.27

Fig. 3.19 (a) Contour plot for the true density distribution of the S1 dataset; (b) a plot of the S1 dataset sampled from (a); (c), (d) and (e) are MDS plots in two dimensions using relative mass for the S1 dataset when $\psi = 4, 16, 64$ and 256, respectively. Best F-measure of ReMass-DBSCAN with different $\psi$ values are also shown. Each colour indicates a true cluster.

Figure 3.20 shows the results of the same experiment of using $\psi = 2, 4, 16, 64$ and 256 on all 20 datasets. I also report the average result over 10 trials and the standard deviation of these trials. It can be seen from the results that ReMass-DBSCAN performs better with $\psi = 4$ in all datasets.




(a) Average F-measure of 20 datasets with the standard deviation on 1 trial.



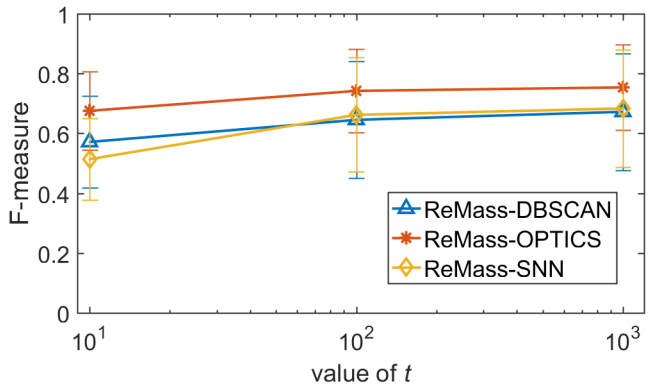

(b) Average F-measure of 20 datasets over 20 trials. The error bars indicate the standard deviation of the average F-measure.

Fig. 3.20 Average F-measure of ReMass-DBSCAN with different $\psi$ values ($t = 1000$) on 20 datasets.

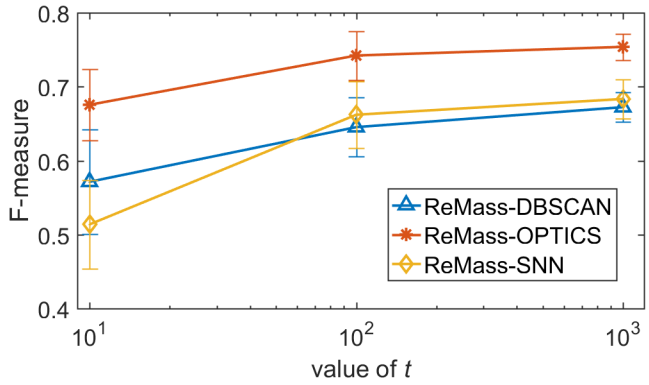
Figure 3.21 shows the effect of $t$ when $\psi = 4$ on all datasets. It is clear that a higher $t$ produces a higher and more stable F-measure with lower variance. This comes at a higher pre-processing cost.

Based on my experimental results, I suggest setting $\psi = 4$ and $t = 1000$ as the default parameter values for the ReMass-DBSCAN as they work well in all datasets.

In summary, all newly introduced parameters are either not sensitive or can be set in a small range.




(a) Average F-measure of 20 datasets with the standard deviation on 1 trial.



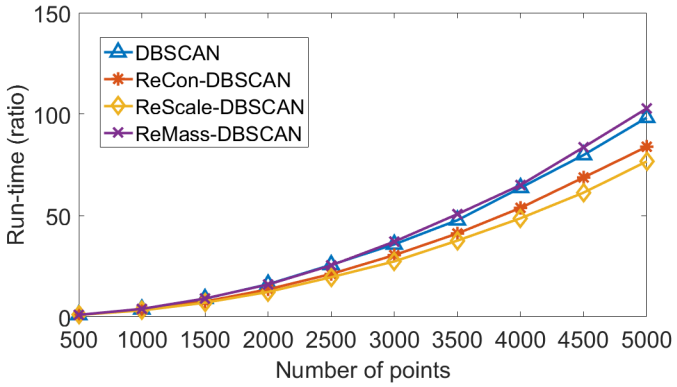

(b) Average F-measure of 20 datasets over 20 trials. The error bars indicate the standard deviation of the average F-measure.

Fig. 3.21 Average F-measure of ReMass-DBSCAN with different $t$ values ($\psi = 4$) on 20 datasets. Error bars indicate the standard deviation of 10 trials.

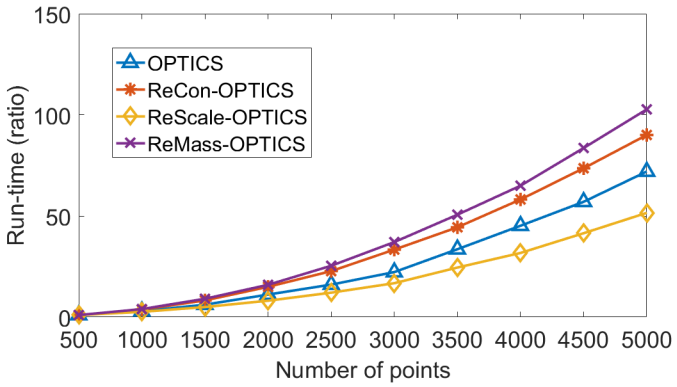
### 3.5.5 Runtime Evaluation

In this section, I evaluate the scalability of the three approaches. Figure 4.11 compares the runtime ratios of different clustering algorithms on a two-dimensional dataset with different data sizes. This result shows that all three approaches retain the time complexity of the density-based algorithm to which they apply.

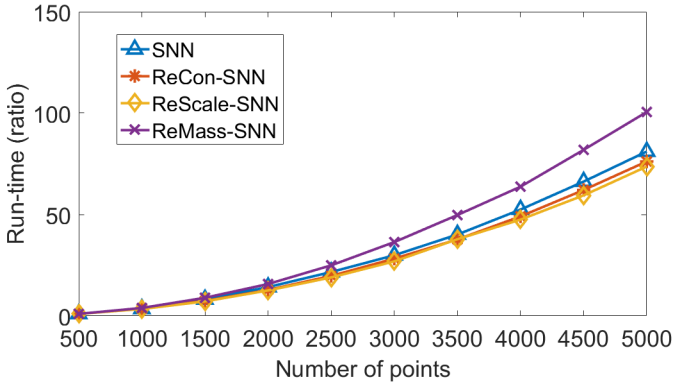
Table 3.8 shows the runtime on the Segment and Spam datasets, with the same parameter settings as used to produce their best F-measure. Notice that SNN takes significantly more time than DBSCAN because of *kNN* matrix calculation. ReMass-DBSCAN takes a longer time to compute the dissimilarity matrix than DBSCAN due to the large number of *iTrees* used. The ReScale approach has linear time complexity and requires an additional small pre-processing time.




(a) DBSCAN and its transformed versions




(b) OPTICS and its transformed versions




(c) SNN and its transformed versions

Fig. 3.22 (a) Contour plot for the true density distribution of the S1 dataset; (b) a plot of the S1 dataset sampled from (a); (c), (d) and (e) are MDS plots in two dimensions using relative mass for the S1 dataset when $\psi = 4, 16, 64$ and 256, respectively. Best F-measure of ReMass-DBSCAN with different $\psi$ values are also shown. Each colour indicates a true cluster.

Table 3.8 Runtime of the two approaches on the two largest datasets: Segment and Pendig (in seconds).

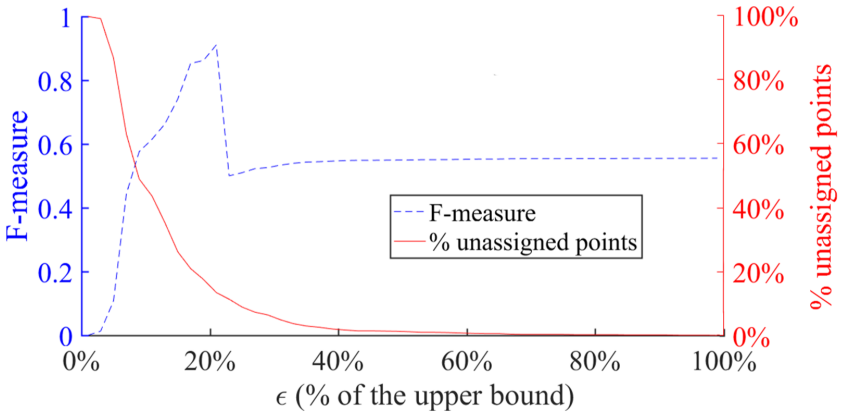
| Algorithm | Segment | Spam |
|---|---|---|
| DBSCAN | 8.4 | 30.7 |
| ReCon-DBSCAN | 8.5 | 30.9 |
| ReScale-DBSCAN | 9.2 | 33.8 |
| ReMass-DBSCAN | 84.8 | 327.8 |
| OPTICS | 8.9 | 31.1 |
| ReCon-OPTICS | 9.1 | 31.3 |
| ReScale-OPTICS | 9.6 | 34.4 |
| ReMass-OPTICS | 85.3 | 328.2 |
| SNN | 29.4 | 122.1 |
| ReCon-SNN | 29.6 | 122.3 |
| ReScale-SNN | 30.1 | 125.4 |
| ReMass-SNN | 105.8 | 419.2 |

## 3.6 Discussion

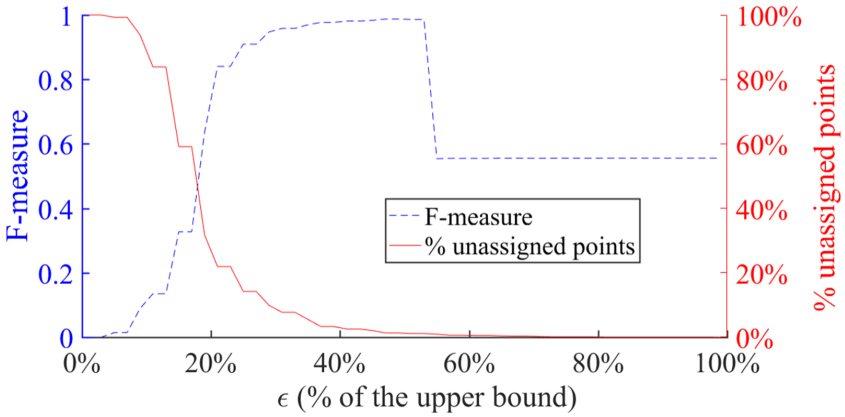
In this section, I provide further discussion about the three approaches in different aspects, such as the brittleness of clustering results, parameter settings and implementations.

### 3.6.1 Brittleness of Clustering Performance

Figure 3.23(a) and Figure 3.23(b) illustrate the F-measure and percentages of points of DBSCAN and ReMass-DBSCAN on the S2 dataset when $\varepsilon$ changes and $MinPts$ is kept fixed. The F-measure value drops when two dense clusters are merged. It can be seen that ReMass enables more data points to be assigned to identified clusters, yielding a higher F-measure. Notice that DBSCAN's best F-measure has 18% unassigned data points, whereas ReMass-DBSCAN's best F-measure has 2% unassigned data points, as shown in Figure 3.23. The reason is that the relative measure transforms the distribution such that all clusters are more concentrated. As a result, a threshold allows more data points in the (originally low-density) cluster to be assigned, as demonstrated in Figure 3.23(a) when compared to Figure 3.23(b). ReMass can also make different cluster modes more distanced from each other. Therefore, ReMass-DBSCAN is less brittle than DBSCAN, i.e., the range of $\varepsilon$ values to achieve good F-measure is wider.




(a) Performance of DBSCAN




(b) Performance of ReMass-DBSCAN

Fig. 3.23 Analysis using the S2 dataset: F-measure and percentages of unassigned points of DBSCAN and ReMass-DBSCAN when $\varepsilon$ changes and $MinPts = 10$. The lower bound of the search range for $\varepsilon$ is the minimum pairwise dissimilarity. The upper bound of the search range for $\varepsilon$ is the minimum value when all points are assigned.

Because the ReMass approach is a dissimilarity measure, it is not suitable for a density-based clustering algorithm, which requires attribute values as input. Grid-based clustering algorithms [25] have such characteristics. In this case, the ReScale approach can be used in order to make performance less sensitive to the cell size setting. See an example in Appendix D.

### 3.6.2 Parameter Settings

The density-ratio computes a local density or conditional density estimated based on the local neighbourhood. As expected, the most important parameter for both the ReCon and ReScale approaches is the $\eta$ used to define the local neighbourhood. I find

that the optimal values for $\eta$ are different for different datasets in terms of obtaining the best clustering results. For a particular dataset, if $\eta$ is less than the optimal value, density-based clustering algorithms may detect multiple artificial modes and split one cluster into a number of subclusters. Generally, $\eta$ only needs to be a small value or slightly larger than $\varepsilon$, e.g., $\eta = 1.1 \times \varepsilon$ for ReCon-DBSCAN and $\eta = 0.1$ for ReScale. The additional parameter $\psi$ in ReScale controls the precision of $\eta$-neighbourhood density estimation. Since a large $\psi$ value takes more computation, I recommend using $\psi = 100$ based on my experiments. I also found that the best $\tau$ is usually between 0.7 to 0.9 for ReCon-DBSCAN and ReCon-SNN on real datasets.

Notice that in one-dimensional space, if $\eta$ is very small or close to 0 then ReScale is similar to a rank transformation [16], which replaces the data by their ranks by labelling rank 1 to the smallest valued data point rank 2 to the second smallest valued data point, and so on.

There are two parameters $\psi$ and $t$ for computing relative mass dissimilarity. My experimental results show that $\psi = 4$ and $t = 1000$ can produce good clustering results for all ReMassed versions of DBSCAN, SNN and OPTICS.

In this thesis, I have searched the parameters in a reasonable range and selected the best performing parameters for each clustering algorithm on various clustering tasks. In practice, it is important to have a heuristic method to automatically set the parameters in each clustering algorithm. Most clustering algorithms have some parameters which require searching; yet most algorithms cannot find clusters automatically without a user making a judgement about the quality of a parameter selection. There are some heuristic methods for setting parameters in DBSCAN, usually based on $k$-nearest neighbour distance as proposed by Ester et al. [22] and Liu et al. [52]. These methods assume that each cluster has a unique density that can be reflected by $k$-nearest neighbour distance distribution. I have tried these methods, but found they cannot perform well on real-world datasets which have clusters with arbitrary densities. Some researchers have employed labelled points to help the DBSCAN detect suitable parameters [44] with a semi-supervised method. A domain expert who understands the data can also be involved to tune the parameters and find meaningful clusters.

However, how to automatically generate these suitable parameters is still an open question.

### 3.6.3 Other Implementations

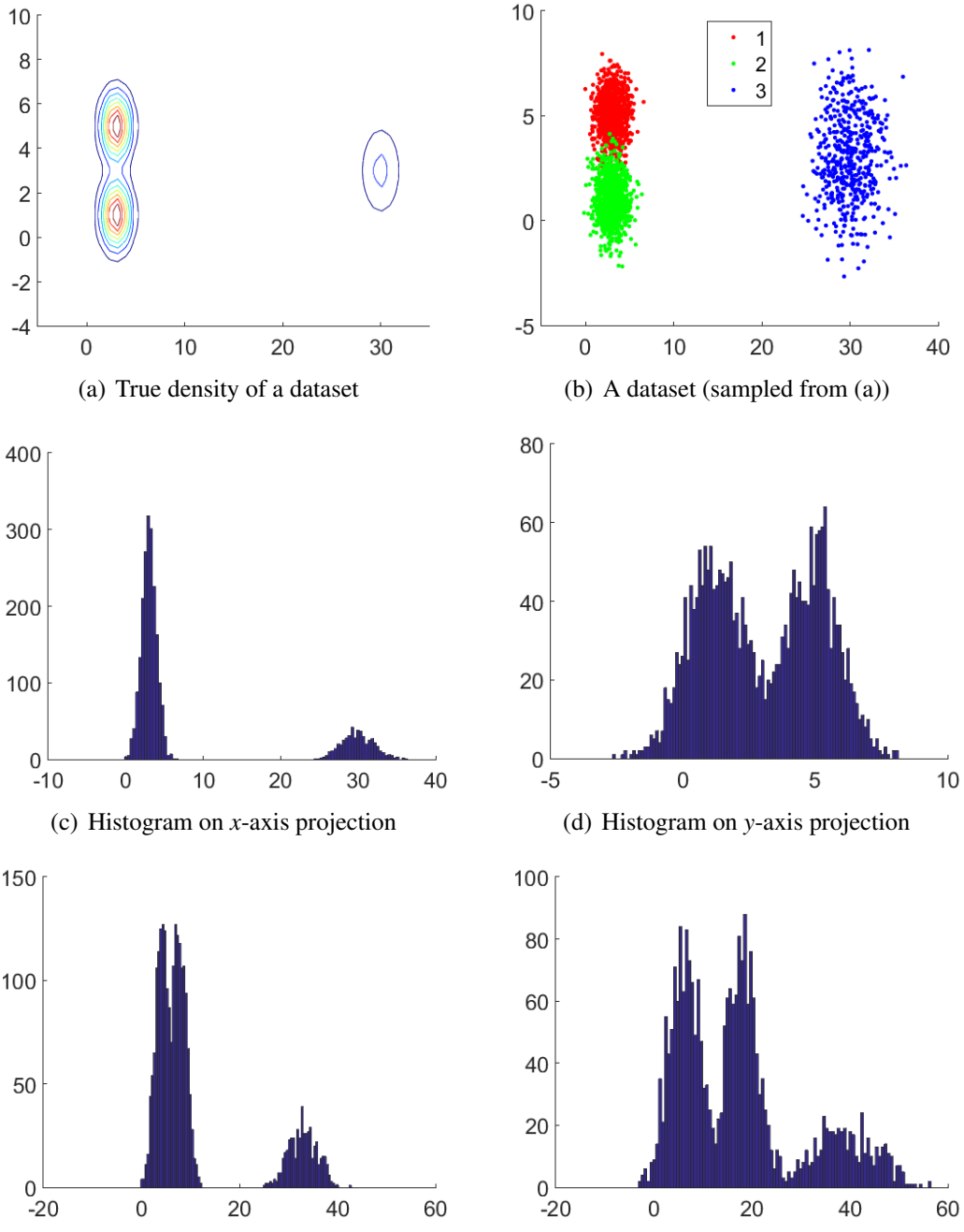
#### ReScale

I have used a simple histogram to estimate the average $\eta$-neighbourhood density in ReScale. A better density estimation method may improve the clustering performance of this approach, albeit possibly come at a higher cost.

ReScale is implemented using the original attributes since I assume that the cluster overlap is not significant on each attribute. If there is a significant overlap between clusters on some attributes, an alternative implementation is recommended to reduce the overlap. I propose to perform multiple (random) linear projections and then do one-dimensional scaling on each projection. Figure 3.24 shows a dataset where rescaling of the $x$ and $y$ axes is less effective due to the cluster overlap, while a random projection can show an non-STS distribution which meets the condition in Equation (3.3) and enhances the effectiveness of ReScale. Figure 3.24(e) and Figure 3.24(f) show two random projections and both illustrate all three clusters with a non-STS distribution. Notice that many random linear projections are required in order to obtain a more stable clustering result.

#### ReMass

Utilising the ReMass approach to overcome the weakness of density-based clustering algorithms is not limited to a specific implementation of estimating the relative mass. In this thesis, *iForest* consists of a set of *iTrees* where each *iTree* is built by using axis-parallel splits. If I employ non-axis parallel splits in each *iTree*, the ReMass-DBSCAN will measure the relative mass distribution of arbitrarily oriented axes. ReMass-DBSCAN using this non-axis parallel splits will perform better on some datasets. Yet another alternative is a different partitioning scheme which produces trees utilising multiple attributes at each internal node, instead of *iForest* [48]. This partitioning




(a) True density of a dataset

(b) A dataset (sampled from (a))

(c) Histogram on $x$-axis projection

(d) Histogram on $y$-axis projection

(e) Histogram on random projection-1

(f) Histogram on random projection-2

Fig. 3.24 (a) Contour plot for the true density distribution of a non-STS distribution dataset; (b) a plot of the dataset sampled from (a); (c) histogram on the $x$-axis projection; (d) histogram on the $y$-axis projection; (e) histogram on a random projection (the projection line is $y = x$); (f) histogram on a random projection (the projection line is $y = 3x$).

process can be performed by using a random linear projection with multiple attributes, and then doing the random split on that projection at each node of a tree, as shown in Algorithm 10 in Appendix A.

The key limitation of the ReMass approach is its high time cost in computing the dissimilarity matrix. This is because a large number of *iTrees* must be generated in order to produce a good estimation of relative mass. Thus, it takes time to access all these *iTrees* to compute each entry in the matrix. My experiment reveals that computing relative mass dissimilarity is 2 to 7 times slower than computing distance. To reduce the runtime, the dissimilarity calculation process can be accelerated by parallel computing as the *iTrees* are independent of each other.

### 3.6.4 Other Issues

When utilising the three approaches to separate all cluster modes, there is a requirement that each cluster mode located at a local maximum-density area be surrounded by local minimum-density areas. If the (estimated) density distribution does not meet this requirement, this measure would not be effective. For example, if clusters with uniform densities are close to each other, the three approaches may not work.

There are many possible non-overlapping groups that can be extracted from a hierarchical model. It is not easy for a user to choose an output from these many possible options. In addition, different hierarchical models make different assumptions when performing agglomeration and division [29]. A linkage method used in a hierarchical model only can find specific shaped clusters [1]. In contrast, my three approaches enable flat models such as DBSCAN to set a global threshold to detect clusters with varied densities. This threshold represents different thresholds in different regions of the space—effectively defining core points in terms of the local density of each region. The proposed density-ratio method also keeps the same advantages of the original density-based algorithm, e.g., ReCon-DBSCAN has the same ability to detect clusters with arbitrary shapes and sizes, and the same computational complexity of DBSCAN.

## 3.7 Summary

In this chapter, I have provided three approaches to overcoming the weakness of density-based clustering algorithms in finding clusters of varied densities. Two approaches enable any DBSCAN-like density-based clustering algorithm to do density-ratio based clustering: the ReCon approach uses the algorithm's density estimator to compute density-ratio, and this only needs a simple modification to the existing density-based algorithm. ReScale is an adaptive scaling approach which operates as a pre-processing step to rescale a given dataset. Then an existing density-based clustering algorithm can be applied to the rescaled dataset. The third approach is a new dissimilarity measure based on relative mass, which is less affected by varied densities across clusters. The ReMass version can be easily constructed for a density-based algorithm by simply replacing the distance measure with this new dissimilarity measure, leaving the rest of the procedure unchanged.

My empirical evaluation shows that all three approaches improve the clustering performance of three existing density-based algorithms, DBSCAN, SNN and OPTICS, in terms of F-measure in synthetic and real-world datasets. These approaches retain the same time complexity of the density-based algorithm to which they apply. Furthermore, ReScale and ReMass can significantly improve the ordering quality in the reachability plots of OPTICS; as a result, clusters can be extracted more effectively using a hierarchical method.

# Chapter 4

# Detecting Clusters in High-dimensional Datasets

*The key to growth is the introduction of higher dimensions of consciousness into our awareness.*

LAO TZU

Biological and medical datasets usually contain tens or hundreds of attributes. Due to the "curse of dimensionality" and irrelevant attributes, traditional clustering algorithms become less effective on high-dimensional datasets. In order to find hidden patterns in these datasets, subspace clustering has been designed as a mechanism to discover clusters in different subspaces.

This chapter provides an answer to Research Question 2. I present an efficient and effective subspace clustering framework (*CSSub*) and propose two algorithms ($CSSub_D$ and $CSSub_I$) based on this framework in order to overcome the weaknesses of existing subspace clustering algorithms.

## 4.1 Motivation

Given a dataset with $d$ attributes, there are $2^d - 2$ different subspaces, which is exponential in the dimensionality of the data. To deal with the large number of possible subspaces, subspace clustering algorithms usually rely on some heuristic

techniques for subspace searches, e.g., top-down search [2] and bottom-up search [3]. All of the subspace clustering procedures perform clustering by measuring the similarity between points in the given feature space, and the subspace selection and clustering processes are tightly coupled.

As discussed in Chapter 2, none of the $k$-medoids and $k$-means type subspace clustering algorithms can detect non-globular clusters, while density-based algorithms avoid this issue. However, most existing density-based subspace algorithms have more than two critical and manually assigned parameters for defining clusters which are very difficult to set in practice. The performance is sensitive to their settings. Therefore, it is important to develop a framework which does not rely on any subspace search strategies or optimisation, and requires fewer user-specified parameters.

I contribute a new subspace clustering framework, named *CSSub* (Clustering of Shared Subspaces), which has the following unique features in comparison with existing methods:

(1) *CSSub* groups points by their shared subspaces. It performs clustering by measuring the similarity between points based on the number of subspaces they share. Existing methods measure similar points based on their similarity in the given feature space, but *CSSub* does not use these features for the similarity calculation.

(2) *CSSub* decouples the candidate subspace selection process from the clustering process. It explicitly splits them into two independent processes. In contrast, existing methods have tightly coupled these two processes.

(3) *CSSub* is flexible, allowing users to employ various methods for defining clusters. It can use different scoring functions to create different types of clusters, e.g., density-based clusters. In contrast, existing methods are unable to easily employ a different cluster definition because of the tightly coupled processes mentioned above. For example, not all cluster definitions satisfy the anti-monotonicity property that enables search space pruning.

I present the subspace clustering framework *CSSub* and propose two subspace clustering algorithms $CSSub_D$ and $CSSub_I$ with the framework as follows.

## 4.2 A New Efficient Subspace Clustering Framework

I first provide definitions for the basic concepts before presenting the proposed subspace clustering framework.

Let $D$ be a dataset of $n$ points of $d$ attributes, represented in the form of a $n \times d$ matrix. A subspace cluster $C_i$ is a submatrix of $D$ with $|s_i| < d$ attributes and $|C_i| \leq |D|$. I denote $S = \{s_1, s_2, ..., s_m\}$ $s_i \subset S$ as the initial set of subspaces to be explored. Let $\pi_s(x)$ denote the point $x \in D$ projected on a subspace $s \in S$. Let the set of $k$ subspace clusters in $D$ be $C = \{C_1, C_2, ..., C_k\}$.

In order to avoid redundancy in this work, I focus on discovering non-overlapping clusters $C$ such that $\forall_{i \neq j, C_i, C_j \in C}\, C_i \cap C_j = \emptyset$.

I now define whether a point $x$ is core in a subspace $s$ by generalising the definition of a core point (Definition 1 in Chapter 3) to any scoring function as:

**Definition 7.** Point $x$ is an **$\alpha$-Core** point in subspace $s$ if $x$ has a high score value based on a scoring function $\alpha(\cdot)$ in $s$, i.e., $(\alpha_s(x) > \tau_s) \leftrightarrow \alpha\text{-}Core_s(x)$. Subspace $s$ is thus a candidate subspace (core-point subspace) for $x$ in which a cluster containing $x$ may exist.

Note that $\tau_s$ depends on subspace $s$; and it is a not a global threshold for all subspaces.

**Definition 8.** Similarity by **shared subspaces**: the similarity between $x_i$ and $x_j$ is defined as the number of shared subspaces in which both $x_i$ and $x_j$ are $\alpha$-Core.

For example, if density is used as the score $\alpha(\cdot)$, then the similarity between $x_i$ and $x_j$ is defined as the number of subspaces in which they are both in dense locations.

Let $A(x) = \{s \in S \mid \alpha\text{-}Core_s(x)\ \}$ be the set of subspaces where point $x$ is an $\alpha$-Core point. As each point is $\alpha$-Core in a set of subspaces which is different from that of another point, the similarity is formulated through normalisation as follows:

$$sim(x_i, x_j) = \frac{|A(x_i) \cap A(x_j)|}{|\, A(x_i) \cup A(x_j)|}$$

Note that the above formulation is the Jaccard similarity [33]; it can be interpreted as the higher the probability that two $\alpha$-Core points are in shared subspaces, the more similar the two points are.

**Definition 9.** Cluster $C$ exists in a set of shared subspaces $S$ if $\forall_{i \neq j,\ x_i, x_j \in C}\ sim(x_i, x_j) \geq \beta_C$, where $\beta_C$ is a threshold for cluster $C$.

Note that $\beta_C$ varies for each cluster and it can be determined automatically if a certain clustering algorithm is employed, e.g., a partitioning-based algorithm $k$-medoids. Hereafter I denote $\alpha$-Core points as "core points".

A conceptual overview of the *CSSub* framework is shown in Figure 4.1. There are four stages in the framework: Stage 1 and Stage 2 are subspace scoring and selection processes; Stage 3 and Stage 4 are clustering by shared subspaces and cluster refinement processes. In Stage 1, I first generate an initial set of subspaces and assess each point to examine whether it is a core point in each of the subspaces. The subspaces are selected as candidate subspaces in Stage 2 if they have core points. In Stage 3, by using a similarity measure based on shared subspaces, such as the Jaccard similarity, an existing clustering algorithm can be employed to perform clustering on the points defined using the candidate subspaces identified in Stage 2. In Stage 4, some method is applied to refine these clusters, e.g., finding the best subspace for each cluster to produce non-overlapping clusters.

## 4.3 Different Types of Clusters

*CSSub* can produce different types of clusters depending on the scoring function $\alpha(\cdot)$ employed. Theoretically, $\alpha(\cdot)$ can be any scoring function. The requirements are that, for any point, its input is the subspace attribute values and its output is a numerical value. I use two different scoring functions to demonstrate the flexibility of the *CSSub* framework, namely, the density score and the isolation path length. They are described in the following three subsections.

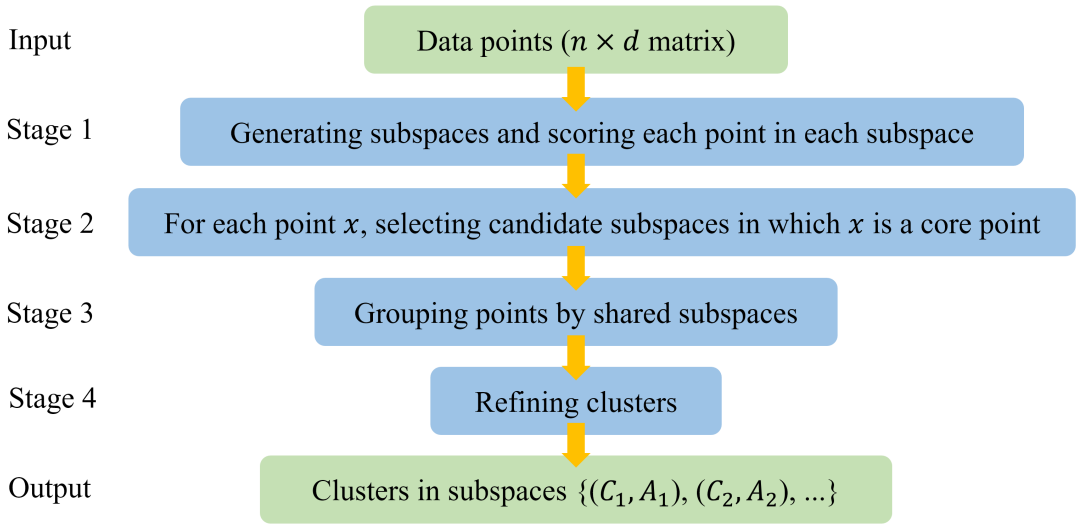



Fig. 4.1 A conceptual overview of the *CSSub* framework for subspace clustering.

### 4.3.1 Density-based Clustering

*CSSub* can produce different types of clusters depending on the scoring function employed. In this paper, I use a density scoring function to demonstrate the ability of *CSSub* to detect arbitrarily shaped clusters.

To use a density function to identify core points, a density estimator is required. I employ the $\varepsilon$-neighbourhood density estimator [22, 36, 37] as follows

$$\widehat{pdf}_s^{\varepsilon}(x) = \frac{1}{nV_d^{\varepsilon}} \sum_j \mathbf{1}(\|\pi_s(x) - \pi_s(x_j)\| \leqslant \varepsilon) \tag{4.1}$$

where $\mathbf{1}()$ denotes the indicator function and $\pi_s(x)$ is point $x \in D$ projected on subspace $s \in S$, and $V_d^{\varepsilon} \propto \varepsilon^d$ is the volume of a $d$-dimensional ball of radius $\varepsilon$. As the volume is constant for all points, it is usually omitted in actual computations.

Since data become sparse in high-dimensional subspaces, a global $\varepsilon_s$ is not appropriate. In order to maximally differentiate points in terms of their estimated density values in a subspace, I use an adaptive $\varepsilon_s$ setting to obtain the maximum variance of the normalised densities such that: $\varepsilon_s = \arg\max_{\varepsilon \in R} \sum_j (p_s^{\varepsilon}(x_j) - \frac{1}{n}\sum_k p_s^{\varepsilon}(x_k))^2$, where $p_s^{\varepsilon}(x) = \widehat{pdf}_s^{\varepsilon}(x) / \sum_y (\widehat{pdf}_s^{\varepsilon}(y))$ is the normalised density of $x$ in subspace $s$ given $\varepsilon$. In my implementation, a fix number $g$ of possible $\varepsilon$ values is examined to obtain the best value $\varepsilon_s$ for each subspace.

Let $p_s^{\varepsilon_s}(x)$ be the density score of point $x$ in subspace $s \in S$. I use $p_s^{\varepsilon_s}(\cdot)$ as the subspace scoring function in stage 1. Subspace $s$ is a selected candidate subspace for $x$ if $(p_s^{\varepsilon_s}(x) > \tau_s)$. Note that $p_s^{\varepsilon_s}(\cdot)$ is used in place of $\alpha_s(\cdot)$ in Definition 7 to determine core points in subspace $s$.

### 4.3.2 Isolation-based Clustering

I introduce an alternative scoring function, i.e., the isolation path length score function, for subspace assessment. The score is derived from the *iForest* (isolation forest) [47, 49], originally used for anomaly detection as the anomaly score.

The intuition is that anomalies are more susceptible to isolation. *iForest* identifies outliers as points (located in sparse regions) having the shortest average path lengths, while normal points (located in dense regions) having longer average path lengths in a dataset. This because outliers can more easily to be separated from the rest of the data.

Because short path lengths indicate anomalies and long path lengths indicate normal points, a cluster can then be defined using neighbouring points which have path lengths longer than a certain threshold.

The advantages of using isolation path length are (1) it is dimensionality unbiased [73]; and (2) it can be computed more efficiently than density estimation.

An *iForest* (consists of multiple *iTrees*) employs a completely random isolation mechanism to isolate every point in the given training set. This is done efficiently by random axis-parallel partitioning (without using a test selection criterion) of the data space in a tree structure until every point is isolated. After building the *iForest*, the isolation path length score can then be computed, and it is defined as follows [49]

**Definition 10.** The **isolation path length** score $L_s(x)$ of a point $x$ in a subspace $s \in S$ is measured as the average path length over a set of $t$ *iTrees* as follows:

$$L_s(x) = \frac{1}{t} \sum_{i=1}^{t} \ell_s^i(x) \tag{4.2}$$

where $\ell_s^i(x)$ is the path length of $x$ in tree $i$ built in $s$.

The path length is the number of nodes of an *iTree* that $x$ traverses in an *iTree* from the root node until the traversal is terminated at a leaf node. The details of the *iTree* building process are provided in Appendix A.

Similar to the density scoring function, I use $L_s(x)$ as the subspace scoring function in Stage 1 if a cluster defined based on path length is used. I use $L_s(\cdot)$ in place of $\alpha_s(\cdot)$ in Definition 7 to determine core points in subspace $s$, such that subspace $s$ is a selected candidate subspace for $x$ if $(L_s(x) > \tau_s)$.

## 4.4 Algorithms in the Subspace Clustering Framework

In this section, I provide the algorithms for the proposed *CSSub* framework shown in Figure 4.1. The steps shown in Algorithm 5 correspond to the stages in Figure 4.1.

**Algorithm 5** *Clustering_by_Shared_Subspaces*($D$, $k$)

**Input:** $D$: input data ($n \times d$ matrix); $k$: number of desired clusters)
**Output:** $C$: a set of clusters in subspaces

1: Generate a set of subspaces $S$ and $V \leftarrow$ *Scoring_Points_in_Subspaces*($D$, $S$)
2: $B \leftarrow$ Sparsifying V
3: $C \leftarrow$ Run clustering algorithm $PAM(B,k)$. The output, for each cluster $C_i \subset C$, has a set of shared subspaces $O_i = \{s \in S \mid \alpha\text{-}Core_s(x) \wedge (x \in C_i)\}$
4: $(C_i, s_i) \leftarrow$ output the subspace $s_i \subset O_i$ which covers the largest number of points in cluster $C_i$ for $i = 1 \ldots k$
5: **return** $(C_i, s_i), i = 1 \ldots k$

The first step in Algorithm 5 generates the initial set of subspaces by enumeration to a maximum subspace dimensionality $d_{max} \geqslant 1$, where $|S| = \sum_{i=1}^{d_{max}} \binom{d}{i} = m < n$. This set of subspaces $S$ is represented using a binary $m \times d$ matrix $V$. Different ways to generate the initial set of subspaces may be used instead.

The subspace scoring algorithm, based on either density or the isolation path length, is shown in Algorithm 6.

In step 2 of Algorithm 5, the output is binary matrix $B$, obtained by sparsifying the matrix $V$. The sparsifying process simply counerts the matrix into a binary value and by $\forall_{i,j}\ (V(i,j) > \tau_s) \leftrightarrow B(i,j) = 1$; otherwise $B(i,j) = 0$, where 1 for $s$ indicates that $s$ is the selected candidate subspace for a point.

**Algorithm 6** *Scoring_Points_in_Subspaces*($D$, $S$)

**Input:** $D$: input data ($n \times d$ matrix); $S$: subspaces ($m \times d$ matrix)
**Output:** $V$: value matrix ($n \times m$ matrix)

```
1: for each point x_i in D do
2:     for each subspace s_j in S do
3:         V(i, j) ← Calculate p^{ε_{s_j}}_{s_j}(x_i) or L_{s_j}(x_i)
4:     end for
5: end for
6: return V
```

Here I set $\tau_s$ to the average score in each subspace as the default parameter, i.e., $\tau_s = \frac{1}{n}\sum_{x \in D} \alpha_s(x)$. Using density as the score, this means that every core point has a density larger than the average density in the subspace. Similarly, using isolation path length, it means that every core point has a path length longer than the average path length in the subspace.

Table 4.1 gives the time and space complexities for the candidate subspace selection process using the two scoring functions. The density scoring function has the same quadratic time complexity of the density estimation function used in existing density-based clustering algorithms due to the pairwise distance calculations. The isolation path length scoring function has linear time complexity.

Table 4.1 Time and space complexities of the candidate subspace selection process based on density score and isolation path length. $m$ is the number of subspaces, $g$ is the number of searches for the best density estimation, $t$ is the number of *iTrees* and $\psi$ is the subsample size.

| Scoring function | Parameter | Time complexity | Space complexity |
|---|---|---|---|
| Density | $\tau_s$ | $O(mgdn^2)$ | $O(dn)$ |
| Isolation path length | $\tau_s$, $\psi$, $t$ | $O(mtn\log(\psi))$ | $O(dn + t\psi)$ |

In step 3 of Algorithm 5, a $k$-medoids algorithm [38] is used for clustering the data in matrix B using the Jaccard similarity as the similarity measure. I use the partitioning around medoids (PAM) algorithm [39], as shown in Algorithm 7.

For each detected cluster $C_i \subset C$ at the end of step 3 in Algorithm 5, it has a set of shared subspaces $O_i = \{s \in S \mid \alpha\text{-}Core_s(x) \wedge (x \in C_i)\}$.

**Algorithm 7** *PAM*($B$, $k$)

**Input:** $B$: core-point subspace matrix ($n \times m$ matrix); $k$: number of desired clusters
**Output:** $C$: set of clusters

1: Randomly select $k$ of the $n$ data points as the medoids
2: Each point in $B$ is assigned to the closest medoid to form clusters $C_i, i = 1, \ldots, k$ based on Jaccard similarity
3: Revise the medoid in each cluster which yields the maximum sum of similarity between each point in the cluster and the medoid, i.e., for each cluster $i$, find medoid $M_i = \arg\max_{y \in C_i} \sum_{x \in C_i} sim(x, y)$
4: Repeat steps 2 and 3 until there is no change in point reassignments
5: **return** $C = \{C_1, \ldots, C_k\}$

In step 4 of Algorithm 5, I refine the clustering result of PAM in order to find the best subspace for each cluster. The final subspace for $C_i$ is the one in $O_i$ which covers the largest number of points in $C_i$, i.e., $s_i = \arg\max_{s \in O_i} |\{x \in C_i \mid \alpha\text{-}Core_s(x)\}|$. The final outputs are the $k$ clusters: $\{(C_1, s_1), (C_2, s_2), ..., (C_k, s_k)\}$.

It is interesting to note that in using PAM to group points by shared subspaces, the algorithm implicitly sets $\beta_{C_i}$ in Definition 9 automatically for each cluster $C_i$ having medoid $M_i$ as follows: $\beta_{C_i} = \min_{x \in C_i} sim(x, M_i)$.

I name the algorithms based on density score and isolation path length scoring functions as $CSSub_D$ and $CSSub_I$, respectively. In practice, both $CSSub_D$ and $CSSub_I$ require only one parameter $k$ to be set manually, as default settings can be used for all other parameters.

## 4.5 Empirical Evaluation

This section presents experiments designed to evaluate the performance of $CSSub_D$ and $CSSub_I$ on various subspace clustering tasks.

### 4.5.1 Experimental Methodology

For comparison systems, I selected a $k$-medoids type subspace clustering algorithm PROCLUS [2], a $k$-means type subspace clustering algorithm FG-$k$-means [13]) and a density-based clustering algorithm P3C [56]. Recall that PROCLUS uses a

top-down subspace search strategy, P3C uses a bottom-up subspace search strategy and FG-$k$-means is a soft subspace clustering algorithm. In addition, I include a full-space clustering algorithm $k$-medoids [38] in order to judge the effectiveness of subspace clustering for real-world tasks. Because it is difficult to assess the clustering performance for algorithms which produce many redundant clusters, this type of subspace clustering algorithm is omitted in the comparison.

### Datasets

I used three new synthetic datasets[1] and the same 18 real-world datasets as used in the last chapter to ascertain the ability of different subspace clustering algorithms in handling high-dimensional data. Table 4.2 presents the properties of the synthetic datasets which have different dimensions.

Table 4.2 Data properties, sorted by dimensionality.

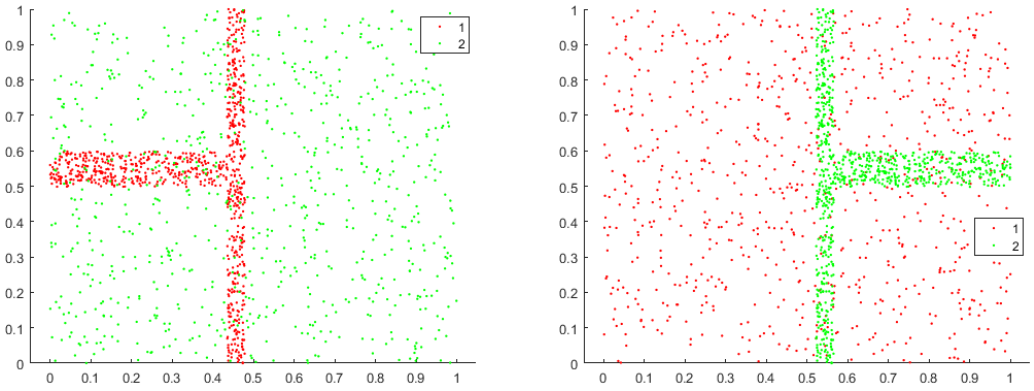
| dataset | $n$ (#points) | $d$ (#Dimensions) | $\varsigma$ (#Clusters) |
|---|---|---|---|
| D50 | 1596 | 50 | 13 |
| S1500 | 1595 | 20 | 12 |
| 2T | 1600 | 4 | 2 |

The 2T synthetic dataset contains two T-shaped clusters embedded in a four-dimensional space, while the S1500 and D50 synthetic datasets have various Gaussian mixtures embedded in different subspaces. Every irrelevant attribute in these synthetic datasets is uniformly distributed between 0 and 1. Figure 4.2 and Figure 4.3 show the data distributions of the 2T and S1500 datasets in two different subspaces, respectively. The D50 dataset has 50 attributes and more irrelevant attributes than the S1500 dataset. S1500 has 12 subspace clusters, 1595 points and 20 attributes.

### Evaluation Measures Used

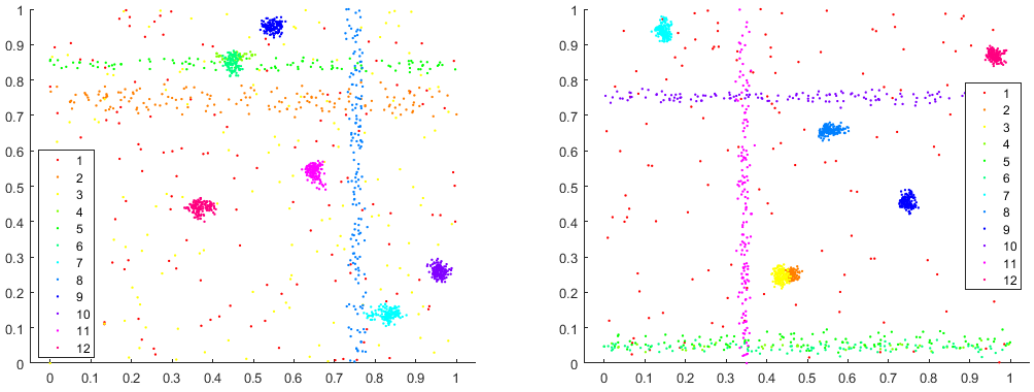
For a fair evaluation, I searched each parameter in a subspace clustering algorithm within a reasonable range or set it to the default setting. For all $k$-means and $k$-medoids type clustering algorithms, I set $k$ to the true number of clusters. Furthermore, I ran

[1] Synthetic datasets S1500 and D50 are from Müller et al. [57].




(a) Data plotted in attribute 1 & 2 (b) Data plotted in attribute 3 & 4

Fig. 4.2 Distributions of the 2T dataset in two subspaces.




(a) Data plotted in attribute 1 & 2 (b) Data plotted in attribute 3 & 4

Fig. 4.3 Distributions of the S1500 dataset in two subspaces.

Table 4.3 Parameters and search ranges for all algorithms used in the experiments.

| Algorithm | Parameters and search range |
|---|---|
| $CSSub_D$ | $k = c$ |
| $CSSub_I$ | $k = c$; $\psi = 256$; $t = 100$ |
| PROCLUS | $k = c$; $avgD = 0.5d$ |
| EWKM | $k = c$; $\gamma = 10$ |
| FG-$k$-means | $k = c$; $\eta = 10$; $\lambda = 10$ |
| LAC | $k = c$; $h = 10$ |
| P3C | $\alpha = 0.001$; *poisson threshold* $\in \{5, 10, ..., 100\}$ |
| $k$-medoids | $k = c$ |

these algorithms over 20 trials using different random seeds for initial $k$ centres in

order to get their best performance. Table 4.3 lists the parameters and their search range for each algorithm.

I evaluate the clustering performance in terms of F-measure, since the ground truth subspaces for each cluster are not available for real-world datasets. Given a clustering result, I calculate the precision score and the recall score for each cluster based on the contingency matrix, and then calculate the F-measure. I report the best clustering performance within a reasonable range of parameter search for each algorithm.

#### Implementation Platform

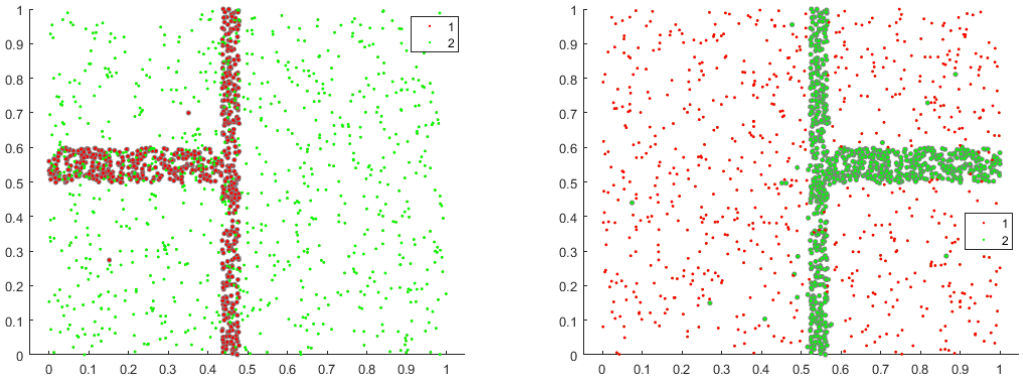
I used the source code from the WEKA platform provided by Müller et al. [57] for PROCLUS and P3C. FG-*k*-means [13] is also implemented in the WEKA platform. $CSSub_D$, $CSSub_I$ and other algorithms used in my experiments are implemented in Matlab. The experiments are run on a machine with four cores (Intel Core i7-3770 3.40GHz) and 24GB memory. All datasets are normalised using the *min-max* normalisation before the experiments begin. Note that each dataset has numeric attributes only, except Dermatology and Ionosphere, where each has one binary attribute in addition to other numeric attributes.

### 4.5.2 Clustering Performance

#### Performance on Synthetic Datasets

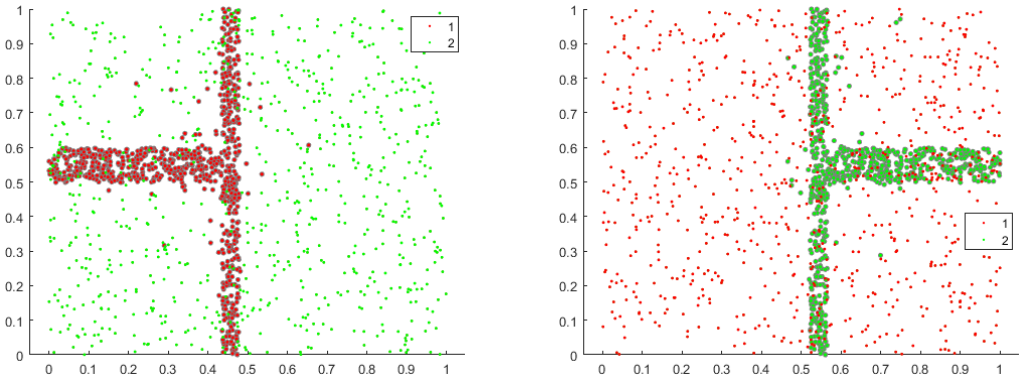
In this section, the 2T synthetic dataset is used to examine whether a subspace clustering algorithm can detect arbitrarily shaped clusters in subspaces of different dimensions.

Figure 4.4 to Figure 4.7 show clusters detected by the four hard subspace algorithms on the 2T dataset. All four algorithms correctly identified the subspace for each cluster on 2T. But the number of correctly assigned points differs greatly among these algorithms. $CSSub_D$ incorrectly assigned a few points only (see Figure 4.4), while $CSSub_I$ incorrectly assigned some points around the high-density areas (see Figure 4.5). Figure 4.6 shows that the clusters detected by PROCLUS have globular



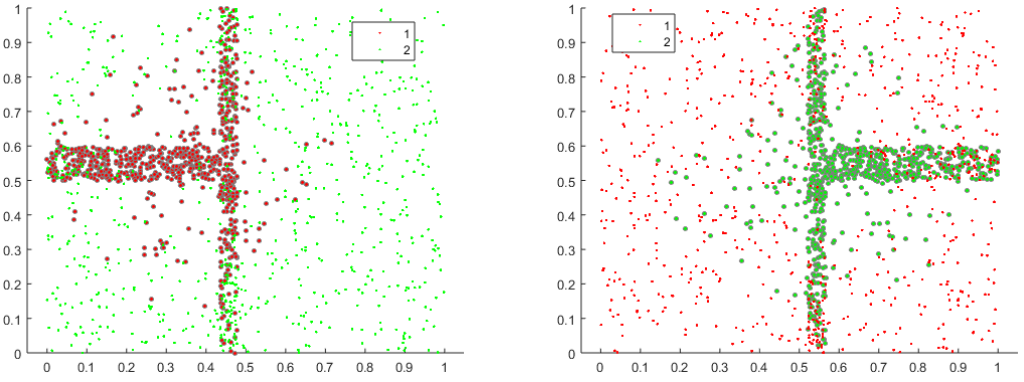

(a) Data plotted on attributes 1 & 2 (b) Data plotted on attributes 3 & 4

Fig. 4.4 Clusters labelled by $CSSub_D$ on the 2T dataset: F-measure=0.94.



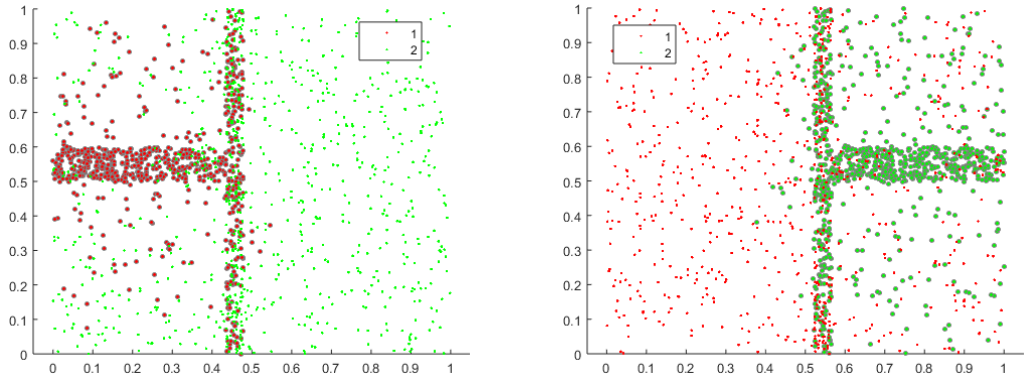

(a) Data plotted on attributes 1 & 2 (b) Data plotted on attributes 3 & 4

Fig. 4.5 Clusters labelled by $CSSub_I$ on the 2T dataset: F-measure=0.92.




(a) Data plotted on attributes 1 & 2 (b) Data plotted on attributes 3 & 4

Fig. 4.6 Clusters labelled by PROCLUS on the 2T dataset: F-measure=0.79.




(a) Data plotted on attributes 1 & 2 (b) Data plotted on attributes 3 & 4

Fig. 4.7 Clusters labelled by P3C on the 2T dataset: F-measure=0.76.

shapes—this is the known weakness of $k$-medoids and $k$-means type algorithms when clustering algorithms are performed on the given feature space—they are unable to detect arbitrarily shaped clusters. Because P3C utilises a bottom-up subspace search strategy, it may incorrectly assign points which have high density in low dimensional projections, as shown in Figure 4.7. The overall result is reflected in the F-measure; $CSSub_D$ and $CSSub_I$ have 0.94 and 0.92, respectively. These results are much higher than those obtained by PROCLUS (0.79) and P3C (0.76).

The relative results for the other two synthetic datasets are similar, i.e., $CSSub_D$ and $CSSub_I$ have better F-measure than PROCLUS, P3C, FG-k-means and k-medoids. These results are summarised in the first row of Table 4.4. The superior clustering performances of $CSSub_D$ and $CSSub_I$ in detecting clusters in subspaces are mainly due to the use of clustering by shared subspaces—the distinctive feature of *CSSub* in comparison with other subspace clustering algorithms. Despite the fact that the same k-medoids algorithm is used in *CSSub*, the use of clustering by shared subspaces has enabled it to detect arbitrarily shaped clusters; this is impossible when k-medoids clustering is performed on attribute values of the given dataset.

### Performance on All Datasets

In this section, I present the clustering results of all six algorithms on all synthetic and real-world datasets. Table 4.4 shows the best F-measure on 21 datasets.

Table 4.4 Best F-measure of different clustering algorithms on 21 datasets; the result of real-world datasets are sorted by F-measure of $k$-medoids. The best two performers on each dataset are in boldface. The last 7 datasets are greyed out because full-space clustering k-medoids have performed well, indicating that they likely have no clusters in subspaces.

| Dataset type | Dataset | Subspace clustering | | | | | full-space clustering |
|---|---|---|---|---|---|---|---|
| | | $CSSub_D$ | $CSSub_I$ | PROCLUS | FG-k-means | P3C | k-medoids |
| Synthetic | D50 | **0.89** | **0.84** | 0.67 | 0.65 | 0.59 | 0.62 |
| | S1500 | **0.84** | **0.89** | 0.75 | 0.70 | 0.66 | 0.68 |
| | 2T | **0.94** | **0.92** | 0.79 | 0.78 | 0.76 | 0.69 |
| Real-world | Glass | **0.46** | **0.43** | 0.40 | 0.37 | 0.28 | 0.33 |
| | Hill | **0.50** | **0.50** | 0.41 | 0.40 | 0.34 | 0.42 |
| | Liver | **0.51** | 0.45 | **0.54** | 0.43 | 0.36 | 0.50 |
| | ILPD | **0.59** | 0.43 | 0.56 | 0.56 | **0.58** | 0.51 |
| | Musk | 0.49 | 0.49 | 0.51 | **0.56** | 0.36 | **0.54** |
| | Sonar | **0.55** | 0.55 | **0.60** | 0.51 | 0.46 | 0.55 |
| | WPBC | 0.50 | **0.58** | 0.43 | 0.48 | **0.60** | 0.58 |
| | PIMA | 0.62 | 0.51 | 0.54 | **0.63** | **0.63** | 0.62 |
| | Segment | 0.49 | 0.46 | 0.43 | **0.61** | 0.28 | **0.68** |
| | Ionosphere | **0.78** | **0.75** | 0.73 | 0.57 | 0.28 | 0.70 |
| | Spam | **0.76** | **0.75** | 0.69 | 0.42 | 0.38 | 0.73 |
| | Thyroid | 0.75 | 0.75 | 0.67 | **0.82** | 0.60 | **0.82** |
| | Seeds | 0.39 | 0.36 | 0.64 | **0.89** | 0.54 | **0.89** |
| | Iris | 0.39 | 0.38 | 0.86 | **0.89** | 0.56 | **0.90** |
| | Wine | 0.71 | 0.49 | 0.88 | **0.92** | **0.93** | 0.91 |
| | Dermatology | 0.70 | 0.73 | **0.86** | 0.46 | 0.56 | **0.92** |
| | WDBC | 0.87 | 0.72 | 0.84 | **0.92** | 0.59 | **0.93** |
| | Breast | 0.88 | 0.76 | 0.90 | **0.95** | 0.90 | **0.95** |

As I do not know whether the real-world datasets have clusters in subspaces, I perform a sanity check: sort the results based on the full-space clustering $k$-medoids algorithm—the datasets in which the full-space clustering has high F-measure are unlikely to have clusters in subspaces. The real-world datasets in Table 4.4 are split into two subsets: one above F-measure = 0.8 and one below.

It is indeed the case that few of the subspace clustering algorithms can perform better than k-medoids full-space clustering in datasets having F-measure $> 0.8$. I can safely assume that the last 7 datasets in Table 4.4 likely have no clusters in subspaces.

I thus focus my comparison hereafter on the datasets in which clusters may exist in subspaces to examine the capability of subspace clustering algorithms.

Among the six algorithms, $CSSub_D$ and $CSSub_I$ are one of the top two performers on 10 and 8 datasets, respectively, out of the first 14 datasets listed in Table 4.4. FG-$k$-means and P3C are one of the top two performers on 3 datasets.

Both $CSSub_D$ and $CSSub_I$ are the best performers and have comparable performance on all 3 synthetic datasets. For the 11 real-world datasets, $CSSub_D$ still performed the best on 7 datasets. $CSSub_I$ performed worse than $CSSub_D$ on 9 datasets.

In the three synthetic datasets, $CSSub_D$ and $CSSub_I$ outperformed $k$-medoids by between 0.16 and 0.27 in F-measure. The best other contender, namely PROCLUS, outperformed $k$-medoids by 0.05 (D50), 0.07 (S1500) and 0.10 (2T).

Out of the 11 real-world datasets, $CSSub_D$ outperformed $k$-medoids by 0.08 or more in F-measure on Glass, Hill, ILPD and Ionosphere datasets, and $CSSub_I$ outperformed on Glass and Hill. P3C performed poorly on most datasets; it performed worse than k-medoids on 10 of out 14 datasets, and nearly half of the points were assigned to noise on Ionosphere.

Note that most subspace clustering algorithms assume that clusters exist in different subspaces. If all clusters can be separated in the full-space only, then they may perform poorly because they prune the subspace search before searching and clustering the full-space [1]. I have observed this phenomenon on the real-world datasets.

It is interesting to see that the performance of the soft subspace clustering algorithm FG-$k$-means is very similar to that of the full-space clustering algorithm $k$-medoids across the datasets shown in Table 4.4. The key difference between soft subspace clustering and full-space clustering is the (dis)similarity calculation, which is based on weighted or unweighted attributes for each cluster. However, the experimental results show that soft subspace clustering still has difficulty detecting subspace clusters when they exist in different subspaces.

I conduct the Friedman test with the post-hoc Nemenyi test [19] to examine whether the differences between any two algorithms are significant in terms of their average

ranks, based on the 14 datasets in which the *k*-medoids performs worse than F-measure of 0.8.

Figure 4.8 shows five subspace clustering algorithms' average ranks and critical differences. This test shows that both $CSSub_D$ and $CSSub_I$ are significantly better than P3C. $CSSub_D$ is also significantly better than FG-*k*-means.

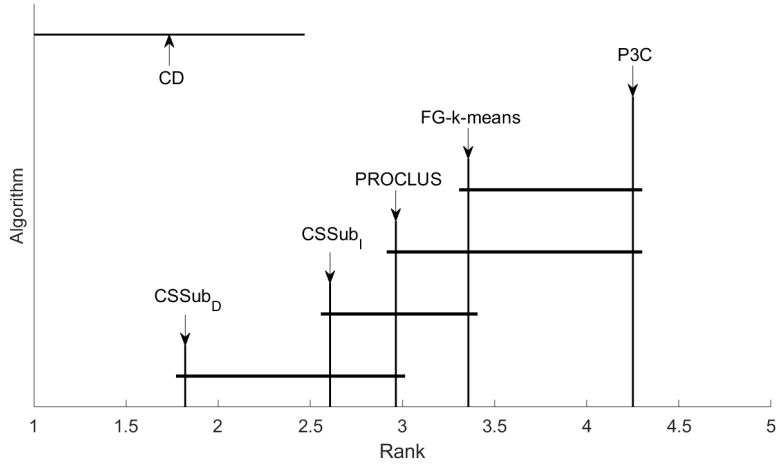


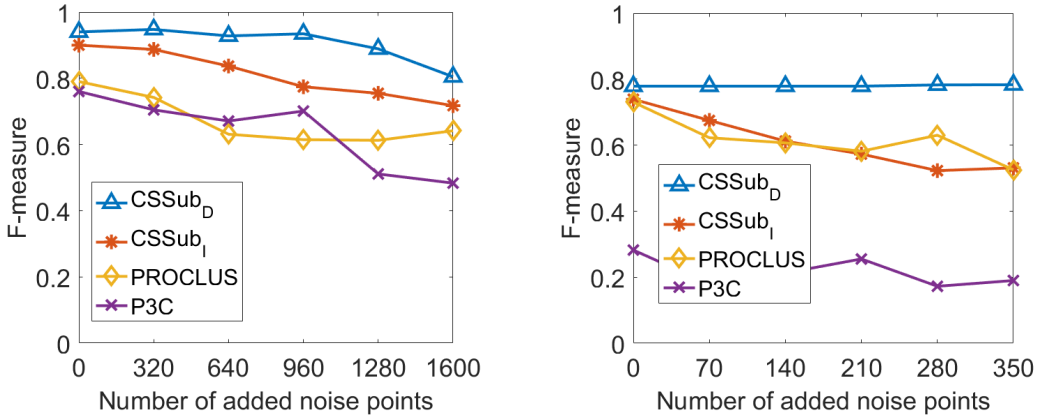

Fig. 4.8 Critical difference (CD) diagram of the post-hoc Nemenyi test ($\alpha = 0.1$). The difference between two algorithms is significant if the gap between their ranks is larger than the CD. There is a line between two algorithms if the rank gap between them is smaller than the CD.

### 4.5.3 Robustness against Noise Points and Noise Attributes

It is important for subspace clustering algorithms to be robust to noise data points and to noise attributes because high-dimensional datasets often contain both. I examine this capability by adding noise points or noise attributes to the 2T and Ionosphere datasets. To compute the F-measure, I evaluated labels assigned to the original points only for each clustering algorithm when adding the noise points.

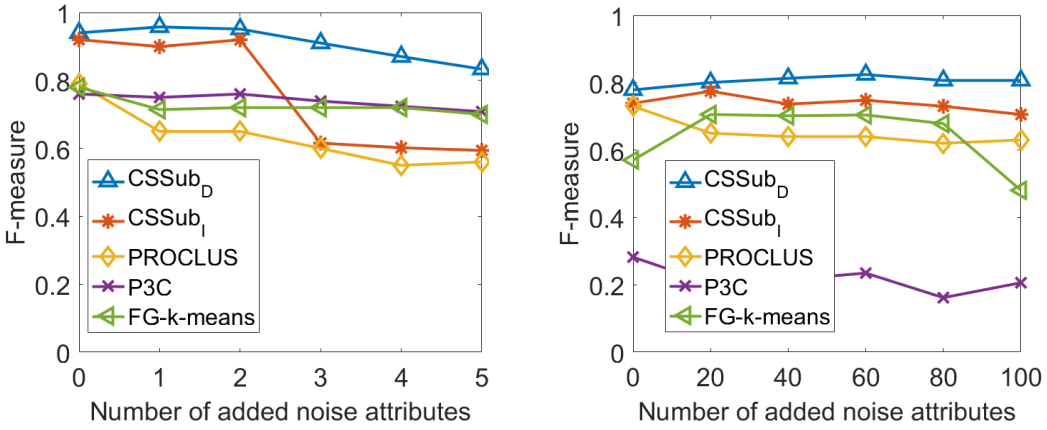
Figure 4.9 shows the F-measure results of $CSSub_D$, $CSSub_I$, PROCLUS and P3C on the 2T and Ionosphere datasets with added noise points. The FG-*k*-means is omitted as it is very sensitive to noise points, similar to *k*-means.

$CSSub_D$ is robust against noise points because it uses an adaptive density threshold for core point identification in each subspace. The noise points were not selected by $CSSub_D$ as core points in most or all of the subspaces. As a result, the added noise




(a) 2T with noise points (b) Ionosphere with noise points

Fig. 4.9 Best F-measure on the 2T and Ionosphere datasets with added noise points. Each noise point has values uniformly distributed in [0,1] for each attribute.




(a) 2T with noise attributes (b) Ionosphere with noise attributes

Fig. 4.10 Best F-measure on the 2T and Ionosphere datasets with added noise attributes. Each noise attribute has values uniformly distributed in [0,1].

points had little or no effect on its clustering result. In contrast, all other algorithms, including $CSSub_I$, were less tolerance to noise points.

Figure 4.10 shows the F-measure results of $CSSub_D$, $CSSub_I$, PROCLUS, FG-$k$-means and P3C on the 2T and Ionosphere datasets with added noise attributes. $CSSub_D$ is the most robust algorithm against noise attributes. $CSSub_I$ is less tolerance to many noise attributes because an *iTree* is likely to select a noise attribute for the split in each node.

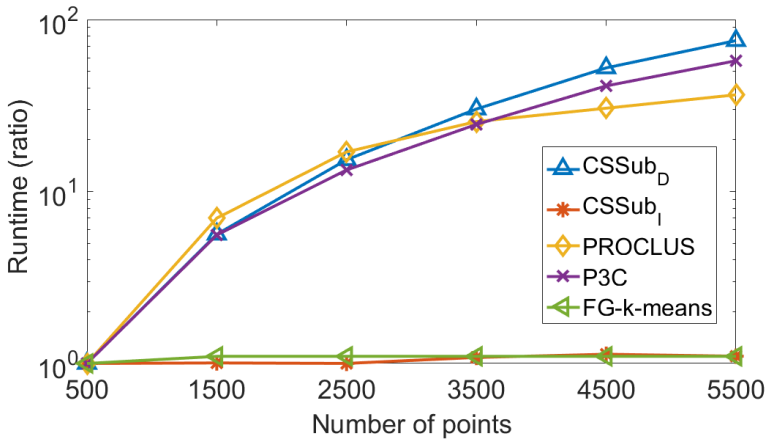



Fig. 4.11 Scalability with respect to the number of points on a synthetic dataset with 4 attributes.

### 4.5.4 Scalability Evaluation

I test the scalability of different subspace clustering algorithms on a synthetic dataset with increasing data size and dimensionality, respectively. For a fair evaluation, I set their parameters to explore the same number of subspaces, i.e., the maximum search depth is equal to the number of dimensions of the dataset used. I manipulate a synthetic dataset with a Gaussian distribution to test the runtime of each algorithm.

#### Increasing Data Size

Figure 4.11 compares the runtime ratios of the five subspace clustering algorithms on the synthetic dataset with increasing data sizes. $CSSub_D$, PROCLUS and P3C show the trend of quadratic time, whereas $CSSub_I$ and FG-$k$-means show a linear time trend.

This experiment shows that the scoring function dominates the runtime of *CSSub* in this dataset. Although PAM (used in *CSSub*) has quadratic time complexity, *CSSub*'s total time complexity is dominated by the subspace assessment, i.e., the scoring function used. The flexibility to use a different scoring function enables *CSSub* to easily convert from one with quadratic time complexity ($CSSub_D$) to one with linear time complexity ($CSSub_I$). Note that for isolation path length scoring, the runtime of building and accessing *iForest* is fixed and fast. After scoring subspaces, the runtime of PAM is only a few seconds, e.g., 2.54 seconds on a subspace of the Spam dataset.

### Increasing Data Dimensionality

Figure 4.12 compares the runtime ratios of the five subspace clustering algorithms on the synthetic dataset with increasing attribute sizes. Both PROCLUS and FG-*k*-means show a linear time trend, while the others show an exponential time trend. This is because the top-down methods only assess individual attributes during each iteration; thus the number of subspace candidates they search is significantly smaller than that of bottom-up methods, which may conduct an exhaustive search in the worst case.

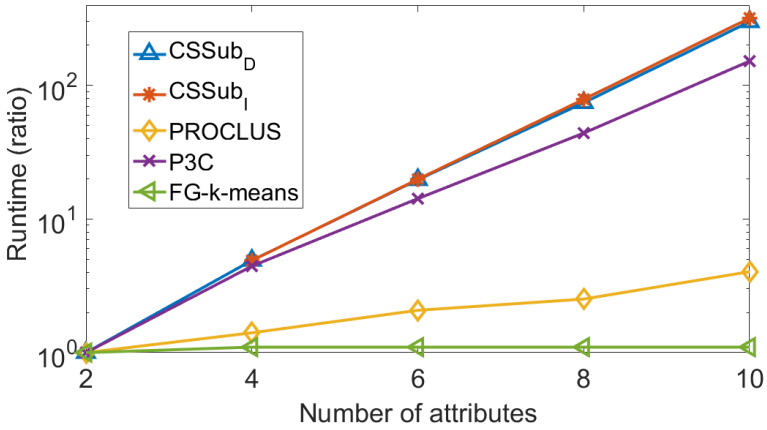



Fig. 4.12 Scalability with respect to the number of attributes on a synthetic dataset with 4500 points.

In practice, *CSSub* has a much lower time complexity because it limits the maximum subspace dimensionality of enumeration. As a result, *CSSub* will exhibit linear time behaviour on a dataset which has dimensionality higher than the number of points, i.e., only one-dimensional subspaces will be assessed, as shown in Figure 4.13.

## 4.6 Discussion

The *CSSub* framework provides a more flexible platform for developing new subspace clustering algorithms which group points by their shared subspaces. The framework allows different combinations of components from either new or existing clustering algorithms. In this thesis, I have demonstrated that $CSSub_D$ and $CSSub_I$ are two effective combinations of an existing PAM clustering algorithm and scoring functions.

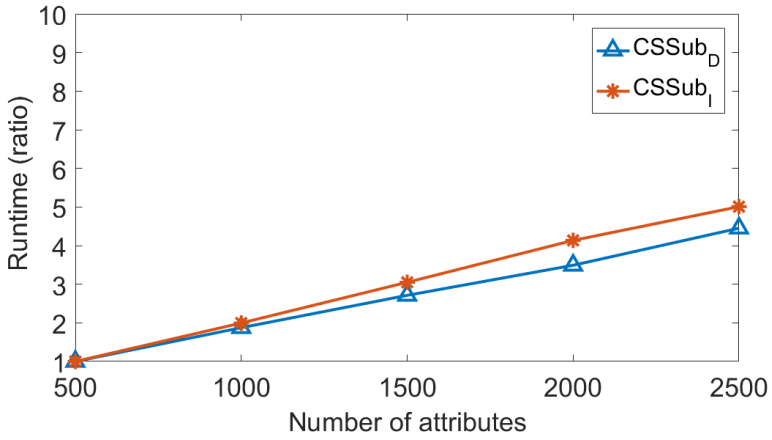



Fig. 4.13 Scalability with respect to the number of attributes on a synthetic dataset with 500 points.

In this section, I discuss additional features of *CSSub*, potential improvements and research directions based on this framework.

### 4.6.1 Other Unique Features

There are two other unique features of *CSSub* which have not been mentioned thus far.

First, recent work [73] has shown that it is inappropriate to use a density score function to assess subspaces in an algorithm which employs a systematic search for subspaces, because a density score is a dimensionality-biased function which biases towards low dimensional spaces. Existing subspace clustering methods that employ a systematic search should use a different criterion to assess subspaces, even though they use a density-based clustering algorithm to perform clustering. For example, ENCLUS employs an entropy-based method for subspace assessment while using density-based clustering. In contrast, *CSSub* enables a density score to assess subspaces because the threshold (i.e., mean of each subspace) employed is equivalent to a relative score which determines core points based on the mean value in each subspace, not an absolute density threshold across all subspaces. The relative score is a kind of normalisation, which can make a biased scoring function become unbiased.

Second, the use of the $k$-medoids or $k$-means in existing subspace clustering methods restricts the output to globular clusters only (e.g., PROCLUS and FG-$k$-means). Yet, *CSSub* has enabled the $k$-medoids to be used to produce arbitrarily

shaped clusters. This is possible because the clustering is performed on points defined by all subspaces under investigation, rather than the given feature space.

### 4.6.2 Subspace Scoring Functions and Threshold Settings

To use a density scoring function, density estimators other than the $\varepsilon$-neighbourhood estimator can be used instead, e.g., kernel density estimation [61]. The key criterion is the time complexity. An example of a linear-time density estimator is LiNearN [75].

To cater for categorical and ordered attributes, I can use other suitable density estimators such as those proposed by Li and Racine [45].

In addition, $\varepsilon$-neighbourhood density estimation is not effective for extremely high-dimensional and sparse datasets. A sparse non-parametric density estimation technique [50] may be used in this case for subspace scoring in *CSSub*.

A similarity measure other than the Jaccard similarity may also be used. The important consideration is that, if the similarity measure is not a metric, then a non-metric clustering algorithm should be used (instead of $k$-medoids) to avoid non-termination problems.

The subspace scoring process can be accelerated by parallel computing, as the assessments are independent of each other.

The parameter $\tau_s$ used in sparsifying the matrix V is an adaptive threshold for identifying core points. I set $\tau_s$ to the average score value in each subspace. I found that the clustering performance on some datasets can be significantly improved when a different $\tau_s$ value is used. Thus, how to automatically obtain a better $\tau_s$ setting for *CSSub* is one of the potential directions for future research.

### 4.6.3 Algorithms for Grouping Points by Shared Subspaces

Although different clustering algorithms can be used here, the $k$-medoids algorithm is chosen because it only requires the similarity matrix $B$ (without requiring access to the

attribute values of the given dataset) and takes only one parameter $k$ (the number of clusters) as input. [2]

Any clustering algorithms which can employ the Jaccard similarity on matrix $B$ can be used to group points by shared subspaces. Thus, clustering algorithms based on $k$-nearest neighbours or $k$-medoids are potential candidates. Other clustering algorithms based on a similarity matrix also can be deployed to group points, e.g., the density-based method DBSCAN [22].

The overall time complexity for the proposed algorithms is still quadratic because the $k$-medoids I used for clustering has a time complexity of $O(n^2)$. There are some methods which can reduce the time complexity of $k$-medoids to linear with little sacrifice of clustering accuracy, such as Clustering LARge Applications (CLARA) [58], which repeatedly performs the PAM algorithm on random subsets of the dataset. Through the sampling, it has powerful scalability and can be used on large datasets.

### 4.6.4 Other Possible Refinements

In this thesis, I assume that only one cluster exists in each of the $k$ subspaces detected. To examine whether the core points detected in a subspace have multiple clusters, I can use a similar method as used by DBSCAN to separate clusters and filter out noise points if they are not connected in that subspace.

The *CSSub* framework can be used to produce either non-overlapping clusters or overlapping clusters. One possible way to produce overlapping clusters is to extract multiple subspaces from the output of PAM which covers most points of each cluster, rather than just the one which covers the maximum number of points in each cluster.

In addition, *CSSub* can identify noise points, which are non-$\alpha$-Core points in all subspaces considered in Stage 2. I found that the clustering performance can be slightly improved on some datasets if all noise points are excluded from clustering in Stage 3.

[2] Unlike the $k$-means algorithm, the $k$-medoids algorithm chooses a point as the centre (medoid or exemplar) for each cluster; thus it is more robust to noise and outliers.

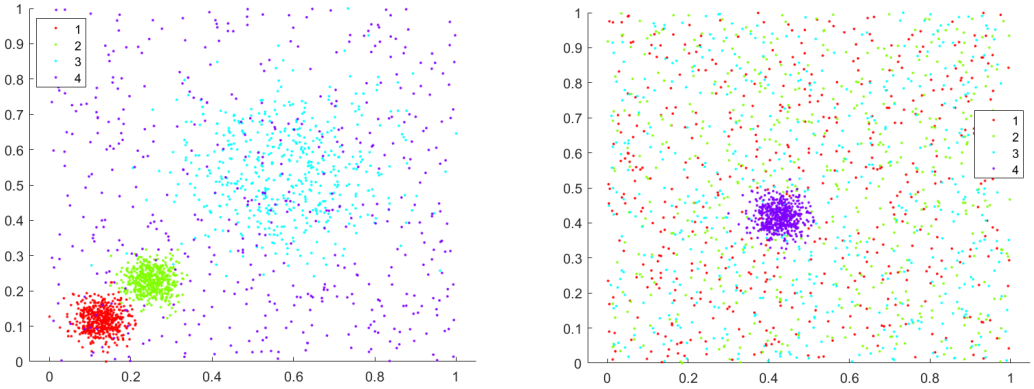
## 4.7 Detecting Subspace Clusters with Varied Densities

It is important to mention that *CSSub* has the ability to detect clusters with varied densities. This is because *CSSub* uses an adaptive threshold $\tau_s$ to identify core points in each subspace $s$. Generally, there are two necessary conditions for *CSSub* to detect all clusters as follows:

- Some points of a cluster (depends on the definition of a cluster) are core points in at least one candidate subspace.
- Different clusters must have different sets of shared candidate subspaces.

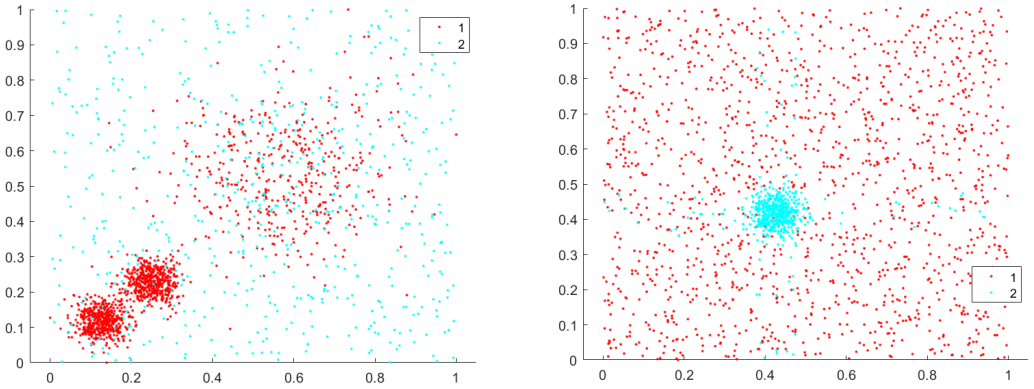
In the last section, I have used synthetic datasets, the S1500 dataset and 2T dataset, under both conditions to demonstrate the ability of *CSSub*. If the first condition is violated, where a cluster doesn't have a core point in any candidate subspace, all data points in the cluster will have similarity of 0 to any other point in the dataset, due to zero shared core-point subspace with others. Then these points may be treated as noise or randomly assigned to other clusters, depending on the clustering algorithm used in Stage 3. When only the second condition is violated, i.e., there are multiple clusters with varied densities detected in one subspace, then the approaches proposed for ansering Research Question 1 can be incorporated in order to separate all clusters in each detected subspace in the refinement Stage 4.

For instance, Figure 4.14 shows a synthetic dataset where three clusters with a non-STS distribution exist in attributes 1 & 2, and another cluster exists in attributes 3 & 4. I first used *CSSub* to identify core points sharing similar subspaces. A subspace cluster in the output can contain multiple clusters with a non-STS distribution. Then I used ReCon-DBSCAN to separate these clusters with varied densities in the same subspace. When setting $k = 2$, $CSSub_D$ can separate the three non-STS distribution clusters from the fourth one, as shown in Figure 4.15. Then ReCon-DBSCAN can be used to separate the three non-STS distribution clusters during the refinement stage. When considering the three non-STS distribution clusters as a whole cluster, $CSSub_D$ can achieve an F-measure of 0.92 on this dataset.




(a) Data plotted in attributes 1 & 2 (b) Data plotted in attributes 3 & 4

Fig. 4.14 Distributions of a dataset having four clusters in two subspaces.




(a) Data plotted in attribute 1 & 2. (b) Data plotted in attribute 3 & 4.

Fig. 4.15 Clusters detected by $CSSub_D$ with $k = 2$ on a 4-dimensional dataset.

In addition, I manipulate two real-world datasets to investigate the ability of *CSSub* to identify clusters from a combination of different types of datasets, and then find clusters with different densities in each datasets. I select the Iris and Seeds datasets for this evaluation because they have no clusters in subspaces, as indicated in Table 4.4. In addition, both of them have three clusters with slightly different densities, shown in Appendix C. The combined dataset has the Iris dataset existing in attributes 1–4, and the Wine dataset existing in attribute 5–11. Irrelevant attributes of each dataset are uniformly distributed in [0,1].

In order to test whether these algorithms can identify clusters from a combination of different type of datasets, I relabel the combined dataset such that all points from

the same original dataset have the same cluster label. This is done for the purpose of evaluation to examine how well they can separate different type of datasets.

Table 4.5 shows the best clustering results of three algorithms on the relabelled combined dataset. Note that I set $k$ in both $CSSub_D$ and PROCLUS to 2. $CSSub_D$ has the highest F-measure of 0.93 and correctly identifies partial relevant attributes for each dataset. PROCLUS and P3C still cannot work well to separate the two datasets from the combined one.

I use ReCon-DBSCAN for the refinement based on the clustering results in Table 4.5 and then calculate the F-measure based on 6 clusters. The final results are shown in Table 4.6. $CSSub_D$ still performs the best with F-measure of 0.75, which is nearly double the F-measure of the other algorithms.

Table 4.5 Clustering results of three subspace clustering algorithms before refinement.

| Algorithm | $CSSub_D$ | PROCLUS | P3C |
|---|---|---|---|
| F-measure | 0.93 | 0.67 | 0.66 |
| Output attributes for Iris dataset | 1 | 1, 2, 3, 10, 11 | 3, 4 |
| Output attributes for Wine dataset | 6, 7, 9 | 5, 6, 8, 9, 11 | 5, 8 |

Table 4.6 Clustering results after refinement.

| Algorithm | $CSSub_D$ | PROCLUS | P3C |
|---|---|---|---|
| F-measure | 0.75 | 0.38 | 0.42 |

In this section, I have shown that *CSSub* has the ability to identify subspace clusters with varied densities, combined with the solution for Research Question 1. In practice, since the ground truth is usually not available, it may not know whether a dataset meets the two necessary conditions for *CSSub* to detect all clusters. Thus, how to set proper parameters for *CSSub* and how to deploy the best approach for refinement are still open questions. Different approaches have different biases. Generally, given an application or a scenario, it requires the domain expert to decide the most meaningful approach for *CSSub* in each stage, based on the task.

## 4.8 Summary

Clusters may exist in different subspaces of a high-dimensional dataset. Traditional full-space clustering algorithms have difficulty in identifying these clusters. Various subspace clustering algorithms use different subspace search strategies. They all require clustering to assess whether one or more cluster(s) exist in a subspace. All of them perform clustering by measuring the similarity between points in the given feature space, and the subspace selection and clustering processes are tightly coupled.

In this chapter, I have provided a subspace clustering framework and proposed two subspace clustering algorithms $CSSub_D$ and $CSSub_I$ based on two different subspace scoring functions. This framework explicitly splits the candidate subspace selection and clustering similar points into two independent processes. It can use different scoring functions to create different types of clusters. I have empirically demonstrated that *CSSub* has the ability to discover subspace clusters with arbitrary shape and size, and has tolerance to noise. If there are multiple clusters with varied densities detected in a subspace, the approach proposed for answering Research Question 1 in Chapter 3 can be incorporated to detect all clusters in the refinement.

# Chapter 5

# Conclusions and Future Work

*Continuous improvement is better than delayed perfection.*

MARK TWAIN

Clustering has become one of the most important processes of knowledge discovery from data in the era of big data. It explores and reveals the hidden patterns in the data, and provides insight into the natural groupings in the data. Clustering is an unsupervised machine learning task that is able to discover clusters automatically. Thus it has been widely used for pattern recognition, information retrieval and image segmentation.

This PhD project aims to solve two problems of density-based clustering in order to efficiently identify the arbitrarily shaped and varied density clusters in high-dimensional data. I have verified the effectiveness of these proposed approaches with extensive empirical evaluation on synthetic and real-world datasets.

## 5.1 Main Contributions of This PhD Project

In this thesis, I have proposed several solutions for overcoming two problems of density-based clustering. In this context, this thesis makes the following contributions.

To answer Research Question 1, I have

(1) identified the condition under which most density-based clustering algorithms fail to find all clusters of differing densities

(2) introduced a less restrictive cluster assumption than that employed in classical density-based clustering; the assumption characterises clusters as regions of locally high density separated by regions of locally low density

(3) proposed two different approaches based on density-ratio to overcome this weakness: (i) a reconditioning approach called ReCon that converts standard density-based clustering algorithms to perform density-ratio clustering; and (ii) a rescaling approach called ReScale that approximates the density-ratio directly from a dataset and enables an existing density-based clustering algorithm, without modification, to apply directly to the rescaled dataset

(4) derived a new neighbourhood function from a data-dependent dissimilarity called ReMass (Relative Mass-based dissimilarity) and verified that it can replace an existing neighbourhood density function to overcome the inability of density-based clustering to find all clusters of varied densities as well

(5) provided an empirical evaluation using DBSCAN [22] and two existing improvement approaches (OPTICS [5] and SNN [21]) in order to demonstrate the effectiveness of ReCon and ReScale.

To answer Research Question 2, I have proposed a new subspace clustering framework named *CSSub* (Clustering of Shared Subspaces). Grouping points by shared subspaces is a unique clustering approach which has (as far as I know) never been attempted previously. The clustering in *CSSub* is performed without examining attribute values in the given dataset. It enables subspace clustering to be conducted (1) without systematic search, and yet it examines the entire space of a systematic search; and (2) by decoupling candidate subspace selection and clustering points into two independent processes—this enables different types of cluster definitions to be employed easily. All of the above features are unique to the proposed *CSSub* framework, compared with existing subspace clustering algorithms. I have proposed two subspace clustering

algorithms $CSSub_D$ and $CSSub_I$ based on two different subspace scoring functions with this framework. I have shown that these two algorithms produce better subspace clustering results on synthetic and real-world datasets in comparison to three existing subspace clustering algorithms (PROCLUS [2], FG-$k$-means [13] and P3C [56]) which employ systematic search or optimisation for subspace searches.

*CSSub* can detect clusters with varied densities existing in different subspaces. If there are multiple clusters with varied densities detected in a subspace, the approach proposed for solving the first problem can be incorporated. Therefore, both problems of density-based clustering can be simultaneously solved in order to efficiently identify the arbitrarily shaped and varied density clusters in high-dimensional data.

With the proposed clustering framework, a domain expert or a data analyst can achieve a better and more meaningful clustering result when handling high-dimensional datasets. I believe this research project will significantly benefit and add to continuing improvement of data-labelling and pattern-recognition tasks.

## 5.2 Future Work

Based on this project, there are some important potential directions for future work:

(1) Investigating the ability of density-ratio based clustering to enhance the performance of different data-mining tasks—in this PhD project, I have provided two approaches based on density-ratio to overcome a weakness of density-based clustering algorithms in finding clusters of varied densities. The density-ratio is a local density or conditional density which is estimated based on local neighbourhoods. It would be possible to apply a similar idea for local anomaly detection, e.g., applying an existing anomaly algorithm to ReScaled data to detect more local anomalies.

(2) Making *CSSub* more scalable and reliable for very high-dimension datasets—in this project, *CSSub* has constrained its search space to a small space of the total search space for high-dimension datasets. Nowadays, an extremely high-dimension dataset usually has hundreds or thousands of dimensions, especially

in image processing and bioinformatics analysis [7]. *CSSub* can be used to only assess individual attributes in these datasets, but is likely to miss potential subspace candidates. Thus, it is important to determine how to select the most interesting subspaces for assessing.

(3) Exploring other possible refinements for *CSSub*—in this project, I only focus on non-overlapping clusters in order to avoid redundancy of subspace cluster results. However, multiview clustering seeks diverse clusterings. It requires some redundancy to keep possibly interesting patterns which might partially overlap with other patterns [9]. I will explore the ability of *CSSub* to produce multiple clustering solutions in the refining stage in the near future.

(4) Designing new external measures for evaluating the performance of density-based clustering algorithms—in this project, I have only used the F-measure as the external evaluation metric. I investigated other external evaluation measures such as purity and normalised mutual information (NMI) [1], and found that their value depends on the points assigned to clusters only, while ignoring all unassigned points, which are labelled as noise. Thus, these other measures can create misleading results for performance comparison between different types of clustering algorithms, e.g., a poor clustering assignment which assigns the majority of the points to noise may result in the highest clustering performance evaluation. Consequently, it is important to develop some external measures for density-based clustering by taking into account noise information.

# References


[1] Aggarwal, C. C. and Reddy, C. K. (2013). Data Clustering: Algorithms and Applications. Chapman and Hall/CRC Press.

[2] Aggarwal, C. C., Wolf, J. L., Yu, P. S., Procopiuc, C., and Park, J. S. (1999). Fast algorithms for projected clustering. In Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, SIGMOD '99, pages 61–72, New York, NY, USA. ACM.

[3] 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, SIGMOD '98, pages 94–105, New York, NY, USA. ACM.

[4] Aitchison, J. (1986). The Statistical Analysis of Compositional Data. Chapman & Hall, London, UK.

[5] Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). OPTICS: Ordering points to identify the clustering structure. In Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, SIGMOD '99, pages 49–60, New York, NY, USA. ACM.

[6] Aryal, S., Ting, K. M., Wells, J. R., and Washio, T. (2014). Improving iforest with relative mass. In Advances in Knowledge Discovery and Data Mining, pages 510–521. Springer.

[7] Assent, I. (2012). Clustering high dimensional data. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2(4):340–350.

[8] Beyer, K. S., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When is "nearest neighbor" meaningful? In Proceedings of the 7th International Conference on Database Theory, ICDT '99, pages 217–235, London, UK, UK. Springer-Verlag.

[9] Bickel, S. and Scheffer, T. (2004). Multi-view clustering. In Proceedings of the Fourth IEEE International Conference on Data Mining, ICDM '04, pages 19–26, Washington, DC, USA. IEEE Computer Society.

[10] Bohm, C., Kailing, K., Kriegel, H.-P., and Kroger, P. (2004). Density connected clustering with local subspace preferences. In Proceedings of the Fourth IEEE International Conference on Data Mining, ICDM '04, pages 27–34, Washington, DC, USA. IEEE Computer Society.

[11] Borah, B. and Bhattacharyya, D. K. (2008). Ddsc: a density differentiated spatial clustering technique. Journal of Computers, 3(2):72–79.

[12] Borg, I., Groenen, P. J., and Mair, P. (2012). Applied Multidimensional Scaling. Springer Science & Business Media.

[13] Chen, X., Ye, Y., Xu, X., and Huang, J. Z. (2012). A feature group weighting method for subspace clustering of high-dimensional data. Pattern Recognition, 45(1):434–446.

[14] Cheng, C.-H., Fu, A. W., and Zhang, Y. (1999). Entropy-based subspace clustering for mining numerical data. In Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 84–93. ACM.

[15] Cherkassky, V. and Mulier, F. M. (2007). Learning From Data: Concepts, Theory, and Methods. John Wiley & Sons.

[16] Conover, W. J. and Iman, R. L. (1981). Rank transformations as a bridge between parametric and nonparametric statistics. The American Statistician, 35(3):124–129.

[17] Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1):21–27.

[18] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society: Series B (Statistical Methodology), pages 1–38.

[19] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research, 7:1–30.

[20] Domeniconi, C., Gunopulos, D., Ma, S., Yan, B., Al-Razgan, M., and Papadopoulos, D. (2007). Locally adaptive metrics for clustering high dimensional data. Data Mining and Knowledge Discovery, 14(1):63–97.

[21] Ertöz, L., Steinbach, M., and Kumar, V. (2003). Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In Proceedings of the 2003 SIAM International Conference on Data Mining, pages 47–58. SIAM.

[22] Ester, M., Kriegel, H.-P., S, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), pages 226–231. AAAI Press.

[23] Ferilli, S., Biba, M., Basile, T. M. A., Mauro, N. D., and Esposito, F. (2008). $k$-nearest neighbor classification on first-order logic descriptions. In Proceedings of the 2008 IEEE International Conference on Data Mining Workshops, ICDMW '08, pages 202–210, Washington, DC, USA. IEEE Computer Society.

[24] Friedman, J., Hastie, T., and Tibshirani, R. (2001). The Elements of Statistical Learning, volume 1. Springer Series in Statistics, Springer, Berlin.

[25] Gan, G., Ma, C., and Wu, J. (2007). Data Clustering: Theory, Algorithms, and Applications (ASA-SIAM Series on Statistics and Applied Probability). Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

[26] Goil, S., Nagesh, H., and Choudhary, A. (1999). Mafia: efficient and scalable subspace clustering for very large data sets. In Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 443–452.

[27] Günnemann, S., Färber, I., Müller, E., Assent, I., and Seidl, T. (2011). External evaluation measures for subspace clustering. In Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pages 1363–1372. ACM.

[28] Guo, G., Wang, H., Bell, D., Bi, Y., and Greer, K. (2003). Knn model-based approach in classification. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", pages 986–996. Springer.

[29] Han, J., Kamber, M., and Pei, J. (2011). Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition.

[30] Hartigan, J. A. and Wong, M. A. (1979). Algorithm AS 136: A *k*-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), 28(1):100–108.

[31] Hinneburg, A. and Gabriel, H.-H. (2007). DENCLUE 2.0: Fast clustering based on kernel density estimation. In Proceedings of the 7th International Conference on Intelligent Data Analysis, IDA'07, pages 70–80, Berlin, Heidelberg. Springer-Verlag.

[32] Hopke, P. K. and Kaufman, L. (1990). The use of sampling to cluster large data sets. Chemometrics and Intelligent Laboratory Systems, 8(2):195–204.

[33] Jaccard, P. (1912). The distribution of the flora in the alpine zone. New phytologist, 11(2):37–50.

[34] Jarvis, R. A. and Patrick, E. A. (1973). Clustering using a similarity measure based on shared near neighbors. IEEE Transactions on Computers, 100(11):1025–1034.

[35] Jing, L., Ng, M. K., and Huang, J. Z. (2007). An entropy weighting *k*-means algorithm for subspace clustering of high-dimensional sparse data. IEEE Transactions on Knowledge and Data Engineering, 19(8):1026–1041.

[36] Kailing, K., Kriegel, H.-P., Kroeger, P., and Wanka, S. (2003). Ranking interesting subspaces for clustering high dimensional data. In PKDD 2003: 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, pages 241–252. Springer.

[37] 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.

[38] Kaufman, L. and Rousseeuw, P. (1987). Clustering by means of medoids. Statistical Data Analysis Based on the L1 Norm and Related Methods, pages 405–416.

[39] Kaufman, L. and Rousseeuw, P. J. (1990). Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons.

[40] Kriegel, H.-P., Kroger, P., Renz, M., and Wurst, S. (2005). A generic framework for efficient subspace clustering of high-dimensional data. In Proceedings of the Fifth IEEE International Conference on Data Mining, ICDM '05, pages 250–257, Washington, DC, USA. IEEE Computer Society.

[41] Kriegel, H.-P., Kröger, P., and Zimek, A. (2009). Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM Transactions on Knowledge Discovery from Data (TKDD), 3(1):1.

[42] Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika, 29(1):1–27.

[43] Kuhn, H. W. (1955). The hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2(1-2):83–97.

[44] Lelis, L. and Sander, J. (2009). Semi-supervised density-based clustering. In 2009 Ninth IEEE International Conference on Data Mining, pages 842–847.

[45] Li, Q. and Racine, J. (2003). Nonparametric estimation of distributions with categorical and continuous data. Journal of Multivariate Analysis, 86(2):266 – 292.

[46] Lichman, M. (2013). UCI Machine Learning Repository.

[47] Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining, pages 413–422. IEEE.

[48] Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2010a). On detecting clustered anomalies using SCiForest. In Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part II, ECML PKDD'10, pages 274–290, Berlin, Heidelberg. Springer-Verlag.

[49] Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2012). Isolation-based anomaly detection. ACM Transactions on Knowledge Discovery from Data (TKDD), 6(1):3:1–3:39.

[50] Liu, H., Lafferty, J. D., and Wasserman, L. A. (2007a). Sparse nonparametric density estimation in high dimensions using the rodeo. In Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, AISTATS 2007, San Juan, Puerto Rico, March 21-24, 2007, pages 283–290.

[51] Liu, H., Zhang, S., Zhao, J., Zhao, X., and Mo, Y. (2010b). A new classification algorithm using mutual nearest neighbors. In 2010 Ninth International Conference on Grid and Cloud Computing, pages 52–57. IEEE.

[52] Liu, P., Zhou, D., and Wu, N. (2007b). VDBSCAN: varied density based spatial clustering of applications with noise. In 2007 International Conference on Service Systems and Service Management, pages 1–4. IEEE.

[53] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, pages 281–297, Berkeley, California. University of California Press.

[54] Mallapragada, P. K., Jin, R., and Jain, A. (2010). Non-parametric mixture models for clustering. In Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), pages 334–343. Springer.

[55] Moise, G. and Sander, J. (2008). Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 533–541. ACM.

[56] Moise, G., Sander, J., and Ester, M. (2008). Robust projected clustering. Knowledge and Information Systems, 14(3):273–298.

[57] Müller, E., Günnemann, S., Assent, I., and Seidl, T. (2009). Evaluating clustering in subspace projections of high dimensional data. Proceedings of the VLDB Endowment, 2(1):1270–1281.

[58] Ng, R. T. and Han, J. (2002). Clarans: a method for clustering objects for spatial data mining. IEEE Transactions on Knowledge and Data Engineering, 14(5):1003–1016.

[59] Niu, X.-x. and Cui, Y.-p. (2011). Improved clustering algorithm based on local agglomerative characteristics. In International Conference on Artificial Intelligence and Computational Intelligence, pages 199–206. Springer.

[60] Parsons, L., Haque, E., and Liu, H. (2004). Subspace clustering for high dimensional data: a review. ACM SIGKDD Explorations Newsletter, 6(1):90–105.

[61] Parzen, E. (1962). On estimation of a probability density function and mode. The Annals of Mathematical Statistics, 33(3):1065–1076.

[62] Patrikainen, A. and Meila, M. (2006). Comparing subspace clusterings. IEEE Transactions on Knowledge and Data Engineering, 18(7):902–916.

[63] Pearson, E. S. (1938). The probability integral transformation for testing goodness of fit and combining independent tests of significance. Biometrika, 30(1/2):134–148.

[64] Procopiuc, C. M., Jones, M., Agarwal, P. K., and Murali, T. (2002). A monte carlo algorithm for fast projective clustering. In Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, pages 418–427. ACM.

[65] Ram, A., Sharma, A., Jalal, A. S., Agrawal, A., and Singh, R. (2009). An enhanced density based spatial clustering of applications with noise. In 2009 IEEE International Advance Computing Conference, pages 1475–1478.

[66] Rodriguez, A. and Laio, A. (2014). Clustering by fast search and find of density peaks. Science, 344(6191):1492–1496.

[67] Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20:53–65.

[68] Sim, K., Gopalkrishnan, V., Zimek, A., and Cong, G. (2013). A survey on enhanced subspace clustering. Data Mining and Knowledge Discovery, 26(2):332–397.

[69] Strehl, A. and Ghosh, J. (2002). Cluster ensembles—a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research, 3(Dec):583–617.

[70] Tan, P.-N., Steinbach, M., and Kumar, V. (2005). Introduction to Data Mining, (First Edition). Addison-Wesley Longman Publishing, Boston, MA, USA.

[71] Ting, K. M. and Wells, J. R. (2010). Multi-dimensional mass estimation and mass-based clustering. In 2010 IEEE International Conference on Data Mining, pages 511–520. IEEE.

[72] Ting, K. M., Zhou, G.-T., Liu, F. T., and Tan, S. C. (2013). Mass estimation. Machine Learning, 90(1):127–160.

[73] Vinh, N. X., Chan, J., Romano, S., Bailey, J., Leckie, C., Ramamohanarao, K., and Pei, J. (2016). Discovering outlying aspects in large datasets. Data Mining and Knowledge Discovery, 30(6):1520–1555.

[74] Wang, W., Zhou, S., and Xiao, Q. (2013). Optimum Vdbscan (o-VDBSCAN) for identifying downtown areas. International Journal of Digital Information and Wireless Communications (IJDIWC), 3(3):66–71.

[75] Wells, J. R., Ting, K. M., and Washio, T. (2014). LiNearN: a new approach to nearest neighbour density estimator. Pattern Recognition, 47(8):2702–2720.

[76] Woollaston, V. (2013). Revealed, what happens in just ONE minute on the internet. http://www.dailymail.co.uk/sciencetech/article-2381188/Revealed-happens-just-ONE-minute-internet-216-000-photos-posted-278-000-Tweets-1-8m-Facebook-likes.html. [Online; accessed 11 January 2017].

[77] Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. IEEE Transactions on Neural Networks, 16(3):645–678.

[78] Zimek, A. and Vreeken, J. (2015). The blind men and the elephant: on meeting the problem of multiple truths in data from clustering and pattern mining perspectives. Machine Learning, 98(1):121–155.

[79] Zitzler, E., Laumanns, M., and Bleuler, S. (2004). A tutorial on evolutionary multiobjective optimization. In Metaheuristics for Multiobjective Optimisation, pages 3–37. Springer.

# Appendix A

# Algorithm to Produce Isolation Path Length Scores

An *iForest* consists of $t$ *iTrees* and each *iTree* is built independently using a subset $\mathcal{D}$, sampled without replacement from $\pi_s(D)$, where $|\mathcal{D}| = \psi$. The maximum tree height is set to $h = \lceil log_2 \psi \rceil$. The parameter $e$ in *iTree* is initialised to 0 at the beginning of the tree building process. The details of the iForest building process can be found in Algorithm 8. The procedure to grow an *iTree* is shown in Algorithm 9. I set $\psi$=256 and $t$=100 as the default parameter for building *iForest*, as suggested by Liu et al [49].

Note that if a leaf node contains more than one points $F \subset \mathcal{D}$, the path length of that leaf node is adjusted by adding $2(\ln|F| + \gamma) - 2$, where is $\gamma \approx 0.58$ is the Euler constant [49].

**Algorithm 8** *iForest*($D$, $e$, $h$)

**Input:** $D$: input data; $t$: number of trees; $\psi$: subsample size
**Output:** *iForest*

1: Initialise *Forest*
2: Height limit $h = \lceil log_2 \psi \rceil$
3: **for** $i = 1$ to $t$ **do**
4: $X \leftarrow sample(\psi, D)$
5: $Forest \leftarrow Forest \cup iTree(X, 0, h)$
6: **end for**
7: **return** *Forest*

**Algorithm 9** $iTree(X, e, h)$

**Input:** $X$: input data; $e$: current height; $h$: height limit
**Output:** an $iTree$

**if** $e \geqslant h$ OR $|X| \leqslant 1$ **then**
    **return** $exNode\{Size \leftarrow |X|\}$
**else**
    Randomly select an attribute $a$ from $X$
    Randomly select a split point $b$ between $min$ and $max$ values of attribute $a$ in $X$
    $X_l \leftarrow filter(X, a < b)$, $X_r \leftarrow filter(X, a \geqslant b)$
    **return** $inNode\{ Left \leftarrow iTree(X_l, e+1, h),$
        $Right \leftarrow iTree(X_r, e+1, h),$
        $SplitAttr \leftarrow a, SplitValue \leftarrow b\}$
**end if**

**Algorithm 10** $iTree'(X, e, h)$

**Input:** $X$: input data; $e$: current height; $h$: height limit; $s$: number of attributes used for a split
**Output:** an $iTree$

**if** $e \geqslant h$ OR $|X| \leqslant 1$ **then**
    **return** $exNode\{Size \leftarrow |X|\}$
**else**
    $A = [a_1, ..., a_s] \leftarrow$ Randomly select $s$ attributes from $X$
    $V = [v_1, ..., v_s] \leftarrow$ Randomly generate $s$ values between 0 and 1
    $M = [m_1, ..., m_s] = tan(\pi V)$
    $X^s \leftarrow$ extract $A$ dimension sub-matrix from $X$
    Randomly select $m_0$ between $min(X^s M^T)$ and $max(X^s M^T)$
    $X_l \leftarrow filter(X, X^s M^T < m_0)$, $X_r \leftarrow filter(X, X^s M^T \geqslant m_0)$
    **return** $inNode\{ Left \leftarrow iTree'(X_l, e+1, h, s),$
        $Right \leftarrow iTree'(X_r, e+1, h, s),$
        $SplitAttr \leftarrow A, SplitValue \leftarrow M, m_0\}$
**end if**

The original *iTree* is built by using axis-parallel splits. Another alternative is a different partitioning scheme which produces trees utilising multiple attributes at each internal node with non axis-parallel splits [48]. This partitioning process can be performed by using a random linear projection with multiple attributes, and then doing the random split on that projection at each node of a tree, as shown in Algorithm 10.

# Appendix B

# Properties of Real-world Datasets

In this research project, I have verified the effectiveness of the proposed approaches with extensive empirical evaluation on real-word datasets in both Section 3.5 and Section 4.5. It is important to understand the properties of datasets extracted from real-life scenarios. All the 18 datasets are from the UCI Machine Learning Repository [46]. The detailed descriptions of these datasets are provided as follows.

- **Iris** This dataset is originally used for plant prediction. It contains 3 classes of 50 instances each with 4 attributes, where each class refers to a type of iris plant. Its attributes include sepal length (in cm), sepal width (in cm), petal length (in cm) and petal width (in cm). The three classes are Iris Setosa, Iris Versicolour and Iris Virginica.

- **Wine** This dataset is wine recognition data, which is 178 results of a chemical analysis of wines grown in Italy from three cultivars. There are 13 attributes including Alcohol, Malic acid, Ash, Alcalinity of ash, Magnesium, Total phenols, Flavanoids, Nonflavanoid phenols, Proanthocyanins, Colour intensity, Hue, OD280/OD315 of diluted wines and Proline.

- **WPBC** This dataset records 198 breast cancer cases by Dr Wolberg since 1984. Each record has 30 input features where the first 30 features are from a digitised image of a fine needle aspirate (FNA) of a breast mass, used to describe characteristics of the cell nuclei present. Other 3 features are Recurrence time

Tumour size (diameter of the excised tumour in centimetres) and Lymph node status (number of positive axillary lymph nodes observed at the time of surgery). There are 151 non-recurring and 47 recurring in the class label. There are 4 cases having a missing value in Lymph node status. Thus, I removed these 4 records from the experiments.

• **Sonar** This dataset has two types of labels including a rock and a mine (metal cylinder). There are 111 records obtained by bouncing sonar signals off a metal cylinder and 97 records of bouncing sonar signals off rocks at various angles and under various conditions. Each record has a st of 60 numbers, representing the energy within a different particular frequency band, integrated over a certain period of time.

• **Seeds** This dataset contains measurements of geometrical properties of kernels which belong to three different varieties of wheat, including Kama, Rosa and Canadian. There are 70 elements for each variety. The internal kernel structure of each element was visualised by a soft X-ray technique and then constructed to 7 real-valued attributes (area, perimeter, compactness, length of kernel, width of kernel, asymmetry coefficient and length of kernel groove).

• **Glass** This dataset is used for classification of types of glass. It contains 7 types of glass with 214 elements, defined in terms of 9 oxide contents including sodium, magnesium, aluminium, silicon, potassium, calcium, barium and iron.

• **Thyroid** This dataset records 215 thyroid disease cases, used to predict whether a patient's thyroid belongs to the class euthyroidism, hypothyroidism or hyperthyroidism. There are five attributes, which are the results of laboratory tests.

• **Liver** This dataset is used to detect liver disorders. There are 345 instances with 6 attributes. The first 5 attributes are all blood tests including mean corpuscular volume, alkaline phosphotase, alamine aminotransferase, aspartate aminotrans-

ferase, and gamma-glutamyl transpeptidase. The last attribute is the number of half-pint equivalents of alcoholic beverages drunk per day.

- **Ionosphere** This is radar data, collected by a system in Goose Bay, Labrador. This system was used to detect free electrons in the ionosphere. There are 351 instances with 34 attributes, summarised into either good or bad radar, depending on whether they show evidence of some type of structure in the ionosphere.

- **Dermatology** This dataset is used for differential diagnosis of erythemato-squamous diseases. It contains 358 patients' records with 12 clinical features and 22 histopathological features. The values of the histopathological features are from an analysis of skin samples under a microscope. All patients are grouped into 6 classes including psoriasis, seboreic dermatitis, lichen planus, pityriasis rosea, cronic dermatitis and pityriasis rubra pilaris.

- **Musk** This dataset is used to predict whether new molecules will be musks or non-musks. There are 166 features describing these molecules based on their shape or conformation. There are 476 instances labelled either musk or non-musk.

- **WDBC** This dataset describes characteristics of the cell nuclei present in breast cancer. It has 569 instances with 30 features, each instance is obtained from a digitised image of a fine needle aspirate (FNA) of a breast mass. There are 357 benign and 212 malignant instances.

- **ILPD** This dataset has 416 liver patient records and 167 non-liver patient records, collected from the northeast of Andhra Pradesh, India. Each record has 10 attributes including age, gender, total bilirubin, direct bilirubin, total proteins, albumin, albumin and globulin Ratio, SGPT, SGOT and alkphos.

- **Breast** This datasets has 699 instances of breast cancer obtained from the University of Wisconsin Hospitals, Madison. It has 10 attributes that are clump thickness, cell size, cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli and mitoses. Each instance has either benign or malignant in its class label.

- **PIMA** This dataset is used to predict based on diagnostic measurements whether a patient has diabetes. It has 768 records with 8 attributes, and all records are from females at least 21 years old of Pima Indian heritage. Attributes include the number of times pregnant, glucose, blood pressure, skinfold thickness, 2-hour serum insulin, body mass index, diabetes pedigree function and age.

- **Hill** This datasets has 1212 records and each record represents 100 points on a two-dimensional graph. It consists of two classes, "Hill" (a bump in the terrain) and "Valley" (a dip in the terrain), based on the plot in order of the Y coordinate.

- **Segment** This dataset is used for image segmentation. There are 2310 instances drawn randomly from a database of 7 outdoor images. Each instance is a 3x3 region image, and is represented by 19 attributes based on the pixel's position, RGB, saturation and hue information.

- **Spam** This dataset is a collection of spam and non-spam emails, and is used for constructing a personalised spam filter. It has 4601 instances with 57 attributes. There are 54 attributes indicating whether a particular word or character is frequently occurring in the email. The other 3 attributes measure the length of sequences of consecutive capital letters.

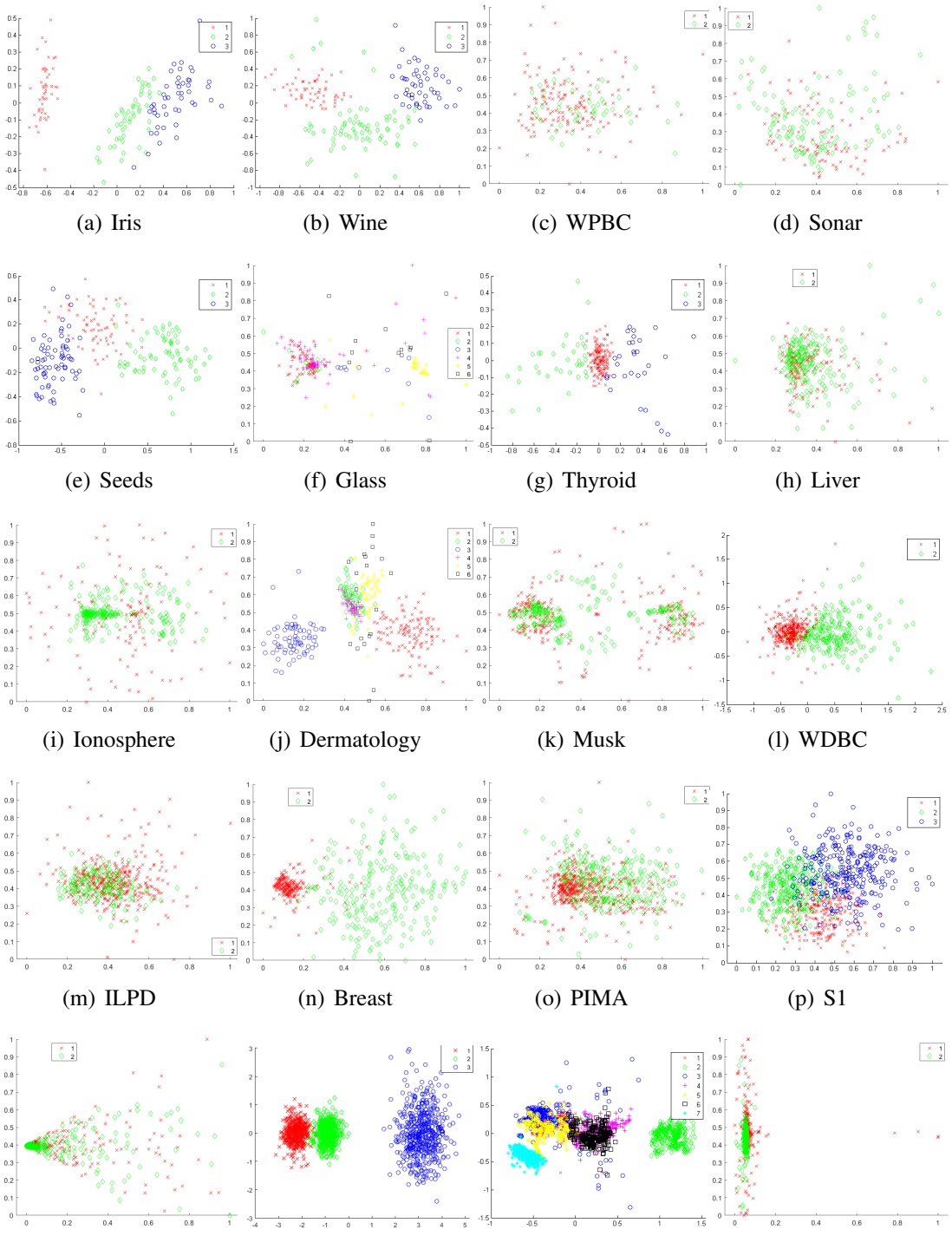
# Appendix C

# Visualisation of ReScale and ReMass

This section uses a visualisation method to displays how far apart data points are from one another before and after applying ReScale and ReMass. The MDS plots for the 20 original datasets with distance measures are shown in Figure C.1. The MDS plots for the 20 rescaled data sets with distance measures are shown in Figure C.2. Figure C.3 shows the MDS plots for the 20 original data using relative mass–based dissimilarity measures. The parameters of ReScale and ReMass are the same settings as used to produce the best F-measure of the converted DBSCAN. In the MDS plot using ReScale and ReMass, the distributions of different clusters appear to become more similar.

Furthermore, to demonstrate the improvement of the ordering quality of the reachability plot of OPTICS, Figure C.4 shows the reachability plots on the 20 original datasets. The reachability plots using ReScale and ReMass are shown in Figure C.5 and C.6, respectively. It is clear that the valleys of ReScale-OPTICS and ReMass-OPTICS are more obvious than those of OPTICS.

Therefore, based on these two approaches, an existing density-based clustering algorithm can perform better than the original in terms of overall clustering performance and the ability to detect clusters with varied densities.



(a) Iris (b) Wine (c) WPBC (d) Sonar

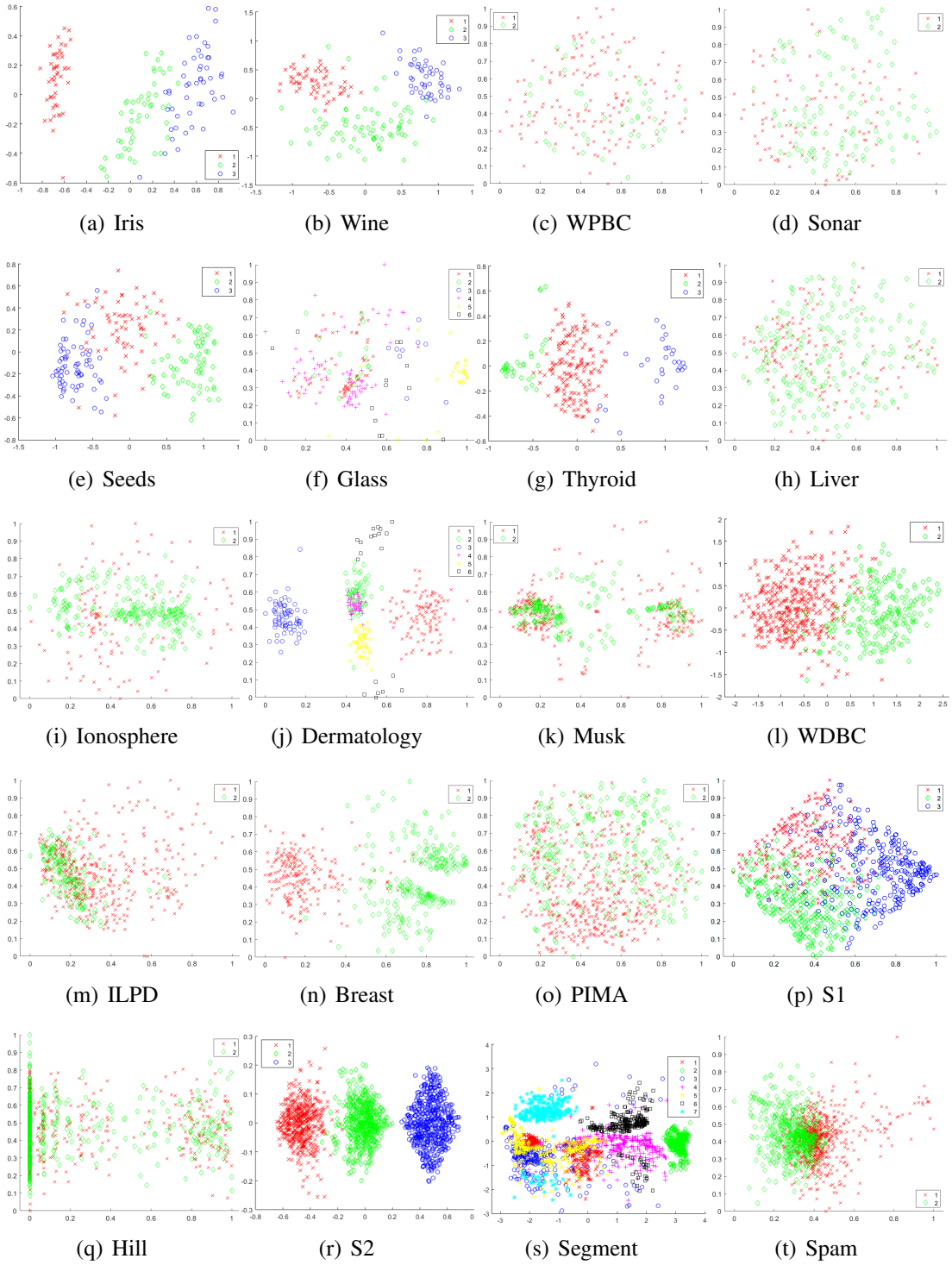
(e) Seeds (f) Glass (g) Thyroid (h) Liver

(i) Ionosphere (j) Dermatology (k) Musk (l) WDBC

(m) ILPD (n) Breast (o) PIMA (p) S1

(q) Hill (r) S2 (s) Segment (t) Spam

Fig. C.1 MDS plots in two dimensions on the original 20 datasets.



(a) Iris (b) Wine (c) WPBC (d) Sonar

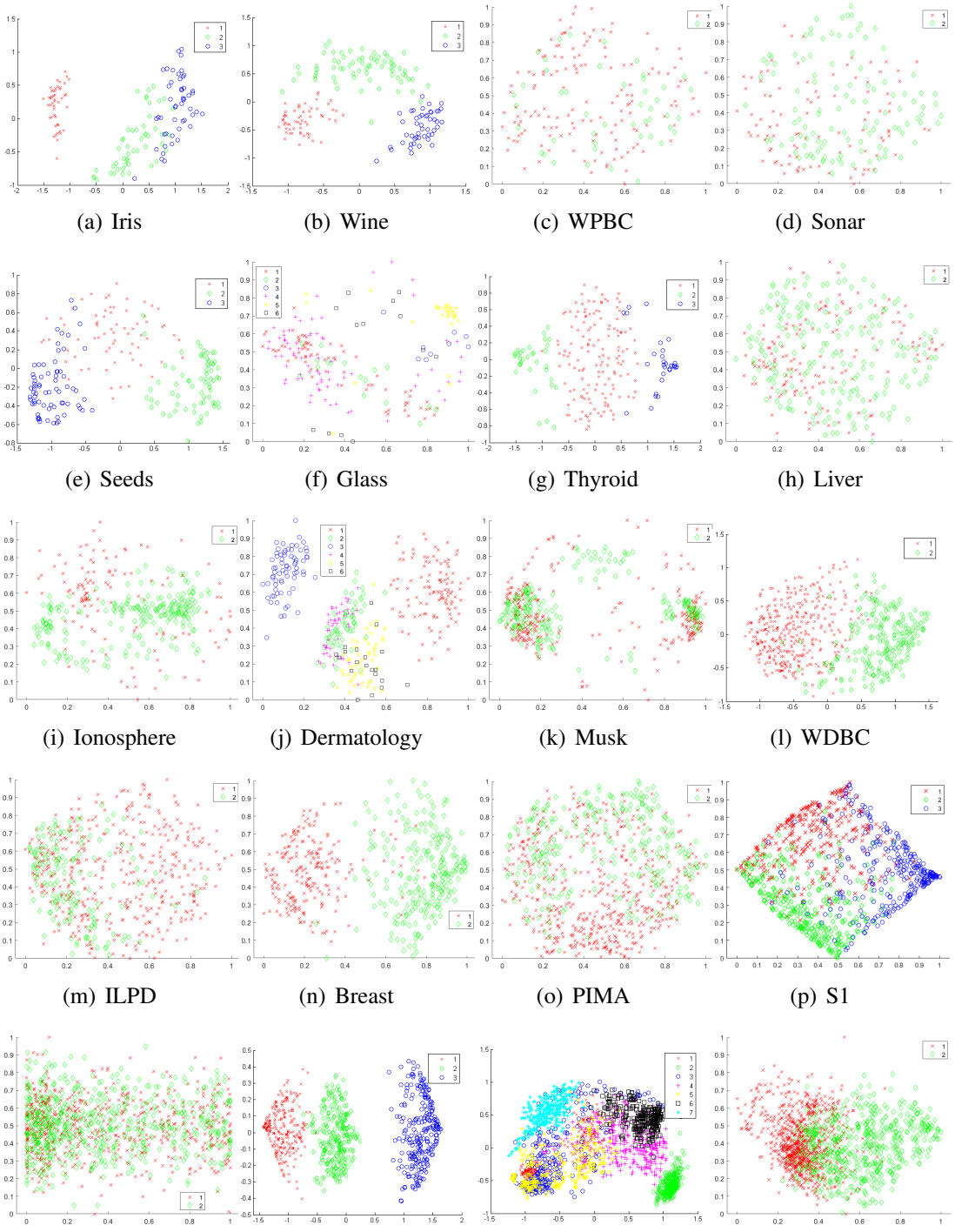
(e) Seeds (f) Glass (g) Thyroid (h) Liver

(i) Ionosphere (j) Dermatology (k) Musk (l) WDBC

(m) ILPD (n) Breast (o) PIMA (p) S1

(q) Hill (r) S2 (s) Segment (t) Spam

Fig. C.2 MDS plots in two dimensions on the ReScaled 20 datasets.



(a) Iris (b) Wine (c) WPBC (d) Sonar

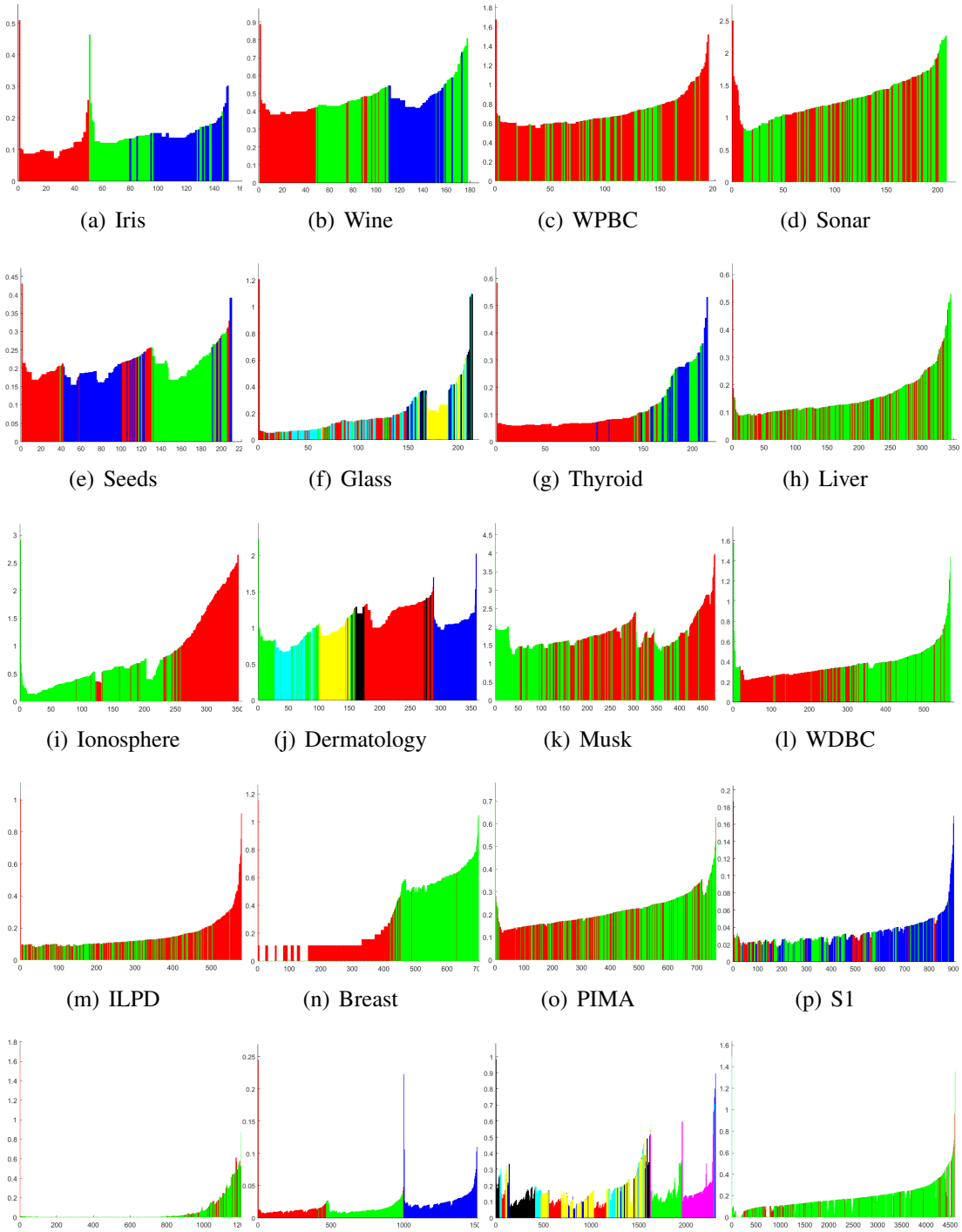
(e) Seeds (f) Glass (g) Thyroid (h) Liver

(i) Ionosphere (j) Dermatology (k) Musk (l) WDBC

(m) ILPD (n) Breast (o) PIMA (p) S1

(q) Hill (r) S2 (s) Segment (t) Spam

Fig. C.3 MDS plots in two dimensions on the 20 datasets using ReMass.



(a) Iris (b) Wine (c) WPBC (d) Sonar

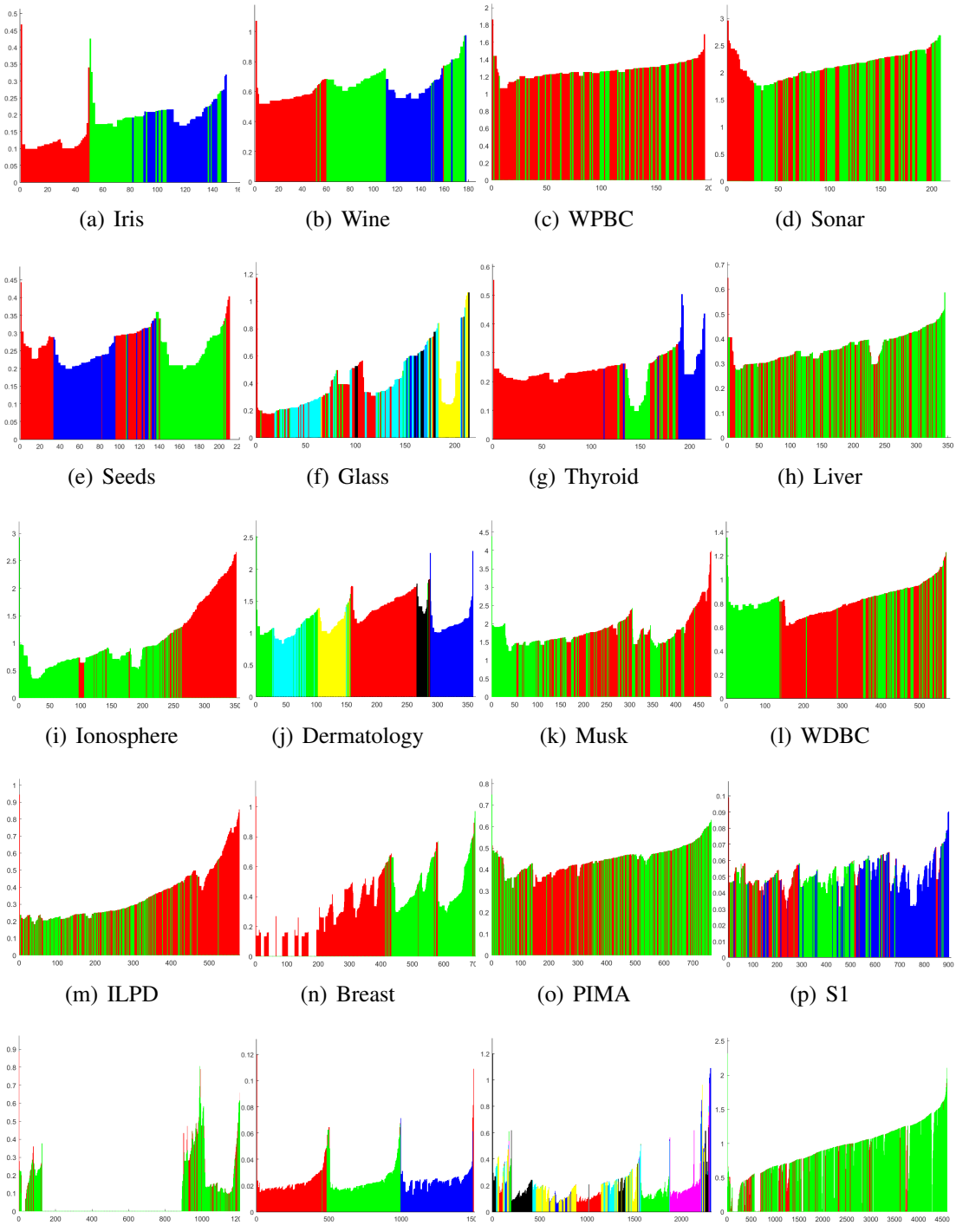
(e) Seeds (f) Glass (g) Thyroid (h) Liver

(i) Ionosphere (j) Dermatology (k) Musk (l) WDBC

(m) ILPD (n) Breast (o) PIMA (p) S1

(q) Hill (r) S2 (s) Segment (t) Spam

Fig. C.4 Reachability plot of OPTICS ($MinPts = 10$) on the original 20 datasets.



(a) Iris (b) Wine (c) WPBC (d) Sonar

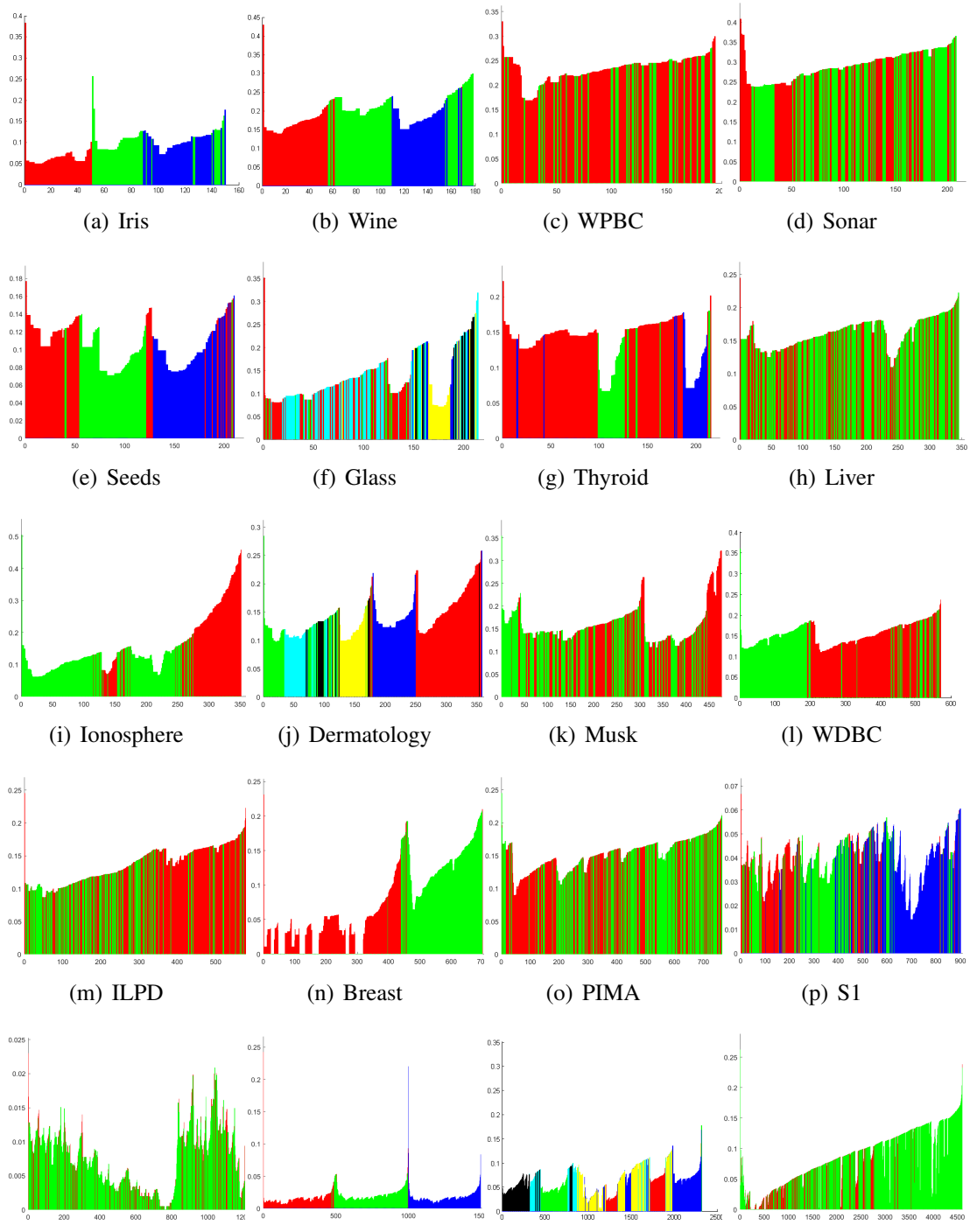
(e) Seeds (f) Glass (g) Thyroid (h) Liver

(i) Ionosphere (j) Dermatology (k) Musk (l) WDBC

(m) ILPD (n) Breast (o) PIMA (p) S1

(q) Hill (r) S2 (s) Segment (t) Spam

Fig. C.5 Reachability plot of OPTICS ($MinPts = 10$) on the ReScaled 20 datasets.



(a) Iris (b) Wine (c) WPBC (d) Sonar

(e) Seeds (f) Glass (g) Thyroid (h) Liver

(i) Ionosphere (j) Dermatology (k) Musk (l) WDBC

(m) ILPD (n) Breast (o) PIMA (p) S1

(q) Hill (r) S2 (s) Segment (t) Spam

Fig. C.6 Reachability plot of OPTICS ($MinPts = 10$) on the 20 datasets using ReMass.

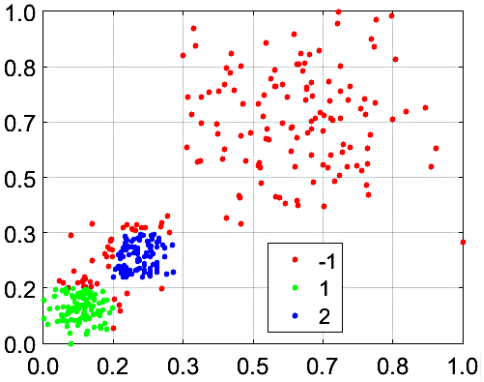
# Appendix D

# Grid-based Clustering

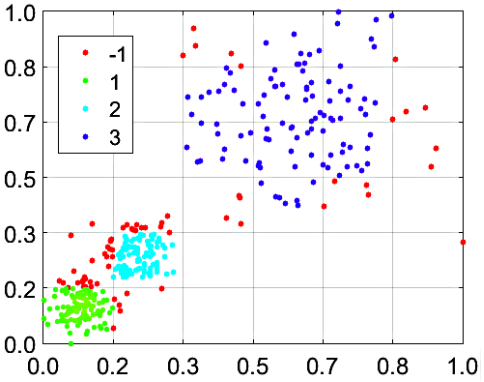
Grid-based clustering is closely related to density-based clustering. It uses grid-like structures to partition the data space into cells, and links adjacent dense cells to form clusters [25]. The main advantage of grid-based clustering is that it is very efficient for large spatial databases. The time complexity normally is $O(nG)$, where $G$ is the total number of cells. However, a major issue is that grid-based clustering cannot be directly applied to a high-dimensional dataset as the number of cells grows exponentially with respect to the number of dimensions and this leads to sparsity of the data. When using grid-based methods in high-dimensional datasets, subspace clustering can be applied to detect clusters on subsets of dimensions [3]. Furthermore, the cell size setting is sensitive to the cluster distribution, i.e., fine cell size may separate a sparse cluster into several subclusters, while coarse cell size may group multiple clusters together.

Figure D.1 illustrates a clustering result using a uniform grid and its ReCon and ReScale results on a non-STS distribution dataset. Note that both the ReCon and ReScale approaches retain the time complexity of the grid-based clustering. The ReCon version ($\eta$ is 2 intervals) can identify more dense cells than the original version, but it has more unsigned points than the ReScale version ($\eta = 0.1$ and $\iota = 1000$). After rescaling, the three clusters have similar density and clusters modes are separated more uniformly. It is clear that the ReCon version of grid-based clustering still suffers from sensitivity to the cell size setting. However, the cell size setting can be more flexible on the ReScaled dataset, thus the ReScale version can obtain a better clustering

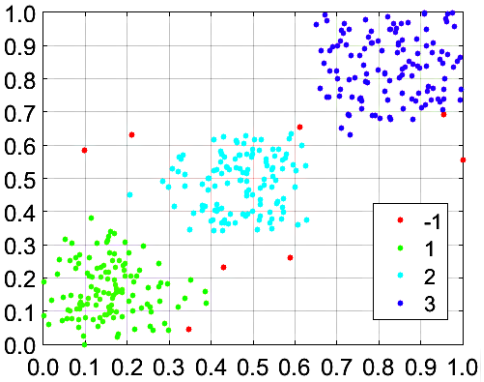
result. Therefore, for large datasets with low dimensions, ReScale with grid-based clustering is recommended.




(a) Grid-based clustering on a non-STS distribution dataset: best F-measure=0.60




(b) ReCon version: best F-measure=0.90




(c) ReScale version: best F-measure=0.99

Fig. D.1 (a) Grid-based clustering on a non-STS distribution dataset; (b) and (c) are the ReCon version and ReScale versions of grid-based clustering on the same dataset, respectively. In order to get the best F-measure, I tuned the number of intervals from 2 to 10 for each dimension, and searched all possible value for other parameters. The "-1" label indicates noise.