Author/s:
Doan, Minh Tuan

Title:
Scalable clustering of high dimensional data in non-disjoint axis-parallel subspaces

Date:
2019

Persistent Link:
https://hdl.handle.net/11343/250810

Scalable clustering of high dimensional data in non-disjoint axis-parallel subspaces

Minh Tuan Doan

ORCID: 0000-0002-0202-9611

Schools of Computing and Information Systems
The University of Melbourne

Submitted in total fulfilment for the degree of
Doctor of Philosophy

November 2020

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

Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Minh Tuan Doan
November 2020

Acknowledgements

First and foremost, I would like to express my sincere gratitude and appreciation to my supervisors who have constantly supported me throughout this journey. Their expertise and passion have inspired me. I have learned from my supervisors not only the research skills but also multiple values that guide me in my future careers. Without my supervisors, it would have been impossible for me to complete the thesis.

I would like to thank my principal supervisor, Professor Christopher Leckie, who provides me with his valuable guidance and constructive feedback for my research, and always makes himself available whenever I seek advice. I would like to thank my co-supervisors, Dr. Jianzhong Qi and Dr. Sutharshan Rajasegarar, for their dedicated support, advice, and comments on my research. I still remember and am deeply thankful for their numerous edits on my papers and thesis.

I wish to further thank my PhD advisory chair, Associate Professor Shanton Chang, for his constructive feedback throughout my research.

I sincerely appreciate Professor Tim Baldwin, Associate Professor Tim Miller and Dr. Sarah Erfani for enriching my PhD experience with different opportunities. They have enabled me to do something I enjoy doing, which is teaching. These experiences have helped me develop multiple skills that I use both in my research and my future career.

I would like to extend my gratitude to my friends, Ben, Gitansh, Han, Lisa, Nick and Pasan, who have added another dimension to my PhD, making it more exciting and less stressful.

Last but not least, I would like to express my love and deep appreciation to my beloved spouse Ha Luong, my parents, Chanh Doan and Cuc Le, and my brother Quang Doan for their love, constant encouragement and support at multiple levels throughout these three years of my PhD. Without them, this memorable journey might not have even started.

Preface

The core chapters of this thesis, Chapters 3 to 7, are based on the following papers. I declare that for each paper I am the main author who has contributed over 50%.

Published Papers

- **Publication 1**: **M. T. Doan**, S. Rajasegarar, M. Salehi, M. Moshtaghi, and C. Leckie. Profiling Pedestrian Distribution and Anomaly Detection in a Dynamic Environment. Proceedings of the 24^{th} ACM International Conference on Information and Knowledge Management (CIKM), 2015, pp. 1827-1830.
- **Publication 2**: **M. T. Doan**, S. Rajasegarar, and C. Leckie. Profiling Pedestrian Activity Patterns in a Dynamic Urban Environment. Proceedings of the 4^{th} International Workshop on Urban Computing (UrbComp), 2015.
- **Publication 3**: **M. T. Doan**, S. Rajasegarar, and C. Leckie. Pedestrian Activity Analysis Using High Dimensional Clustering in a City Environment. Proceedings of the 23^{rd} ITS World Congress on Intelligent Transport Systems, 2016.
- **Publication 4**: **M. T. Doan**, J. Qi, S. Rajasegarar, and C. Leckie. Scalable Bottom-up Subspace Clustering using FP-Trees for High Dimensional Data. Proceedings of the 6^{th} IEEE International Conference on Big Data (IEEE BigData), 2018.

Abstract

Clustering is the task of grouping similar objects together, where each group formed is called a cluster. Clustering is used to discover hidden patterns or underlying structures from the data, and has a wide range of applications in areas such as the Internet of Things (IoT), biology, medicine, marketing, business, and computing. Recent developments in sensor and storage technology have led to a rapid growth of data, both in terms of volume and dimensionality. This raises challenges for existing clustering algorithms and led to the development of subspace clustering algorithms that cope with the characteristics, volumes, and dimensionality of the datasets that are now available. In this thesis, we address the challenges of finding subspace clusters in high dimensional data to achieve subspace clustering with high quality and scalability.

We provide a comprehensive literature review of existing algorithms, and identify the open challenges in subspace clustering of high dimensional data that we address in this thesis, namely: devising appropriate similarity measures, finding non-disjoint subspace clusters, and achieving scalability to high dimensional data. We further illustrate these challenges in a real-life application. We show that clustering can be used to construct a meaningful model of the pedestrian distribution in the City of Melbourne, Australia, in low dimensional space. However, we demonstrate that the clustering quality deteriorates rapidly as the number of dimensions (pedestrian observation points) increase. This also serves as a motivating example on why subspace clustering is needed and what challenges need to be addressed.

We first address the challenge of measuring similarity between data points, which is a key challenge in analyzing high dimensional data that directly impacts the clustering results. We propose a novel method that generates meaningful similarity measures for subspace clustering. Our proposed method considers the similarity between any two points as the union of base similarities that are frequently observed in lower dimensional subspaces. This allows our method to first search for similarity in lower dimensional subspaces and aggregate these similarity values to determine the overall similarity. We show that this method can be applied for measuring similarity based on distance, density, and grids, which enables our similarity measurements to be used with different types of clustering algorithms, i.e., distance-based, density-based, and grid-based clustering algorithms.

We then use our similarity measurement to build a subspace clustering algorithm that can find clusters in non-disjoint subspaces. Our proposed algorithm follows a bottom-up strategy. The first phase of our algorithm searches for base clusters in low dimensional subspaces. Subsequently, the second phase forms clusters in higher dimensional subspaces by aggregating these base clusters. The novelty of our proposed method is reflected in both phases. First, we show that our similarity measurement can be integrated in a subspace clustering algorithm. This not only helps prevent the false formation of clusters, but also significantly reduces the time and space complexity of the algorithm by pruning irrelevant subspaces at an early stage. Second, our algorithm transforms the common sequential aggregation of base clusters into a problem of frequent pattern mining. This enables efficient formation of clusters in high dimensional subspaces using FP-Trees. We then demonstrate that our proposed algorithm can outperform traditional subspace clustering algorithms using bottom-up strategies, as well as state-of-the-art algorithms with other clustering strategies, in terms of clustering quality and scalability to large volumes and high dimensionality of data.

Subsequently, we evaluate the ability of our proposed subspace clustering algorithm to find clusters in datasets from different real-life applications. We conduct experiments on datasets from three different applications. First, we apply our proposed clustering algorithm to pedestrian measurements in the City of Melbourne, and construct a meaningful model that describes the profiles of the distributions of pedestrians that correspond to pedestrian activities at different times of the day. We then use our clustering algorithm to analyze the impacts of a major change in the public transport of Melbourne on the activities of pedestrians. In the second application, we evaluate the ability of the proposed algorithm to work with very high dimensional data. Specifically, we apply our algorithm on ten gene expression datasets, which comprise up to 10,000 dimensions. Next, we explore the ability of our algorithm to produce clustering results that can be used as an intermediate step that assists the construction of a more complicated model. Specifically, we use our clustering result to build an ensemble classification model, and show that this model improves the accuracy of predicting the car parking occupancy in the central business district (CBD) of the City of Melbourne.

By applying our proposed methods in a wide range of applications on datasets with different sizes and dimensionalities, we demonstrate the ability of our algorithm to cluster high dimensional datasets that possess complex structures with high levels of noise and outliers to produce meaningful clustering results that can have practical impact.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Clustering is a task of grouping objects or points based on some similarity measure, where each group is called a *cluster*. Clustering has the ability to discover hidden patterns, or underlying structures from the data. It has a wide range of applications in a variety of areas. For example, it can be used in marketing and business to group customers into different clusters depending on their demographics, gender and product preferences, in order to design tailored business plans that target customers effectively [236]. Another clustering application is in the management of smart cities. For example, sensors have been deployed across the City of Melbourne, Australia, to monitor traffic volumes on the streets and at intersections. Clustering can be used to group locations that observe similar traffic patterns, which subsequently leads to a profile of the traffic distribution in the city. Such a model can provide valuable insights on the design and operation of the transport system of the city. For example, it can reveal the traffic hot spots and the routes that observe similar patterns of increasing traffic volumes during peak hours, which offers insights on road usage, vehicle movement, inbound and outbound flows of traffic to and from the city. In addition, clustering is widely used in different fields of the life sciences. Specifically, in biology, clustering is used to study plant and animal ecology [56], cluster human genes and analyze gene sequences [34, 75]; in medicine, clustering is used to segment intensity-modulated radiotherapy (IMRT) [24] and to analyze patterns of antibiotic resistance [225]. Further, in computer science, different applications of image segmentation [68], recommender systems [58], and anomaly detection [148] rely on clustering to group similar points into the same clusters.

Clustering was first proposed in 1938 [32] and is a well established research area. Recent development in sensor and storage technology has resulted in huge amounts of data becoming available, both in terms of volume and dimensionality. Studies [188, 206, 229] have shown that the growth of data is outpacing the analysis capacity, and there is a strong demand to

extract useful knowledge from this data. These datasets raise novel challenges for existing clustering algorithms, and motivate the development of novel clustering algorithms to cope with the characteristics of data of high volume and dimensionality in order to maintain the cluster quality. This has led to the proliferation of subspace clustering algorithms.

Subspace clustering aims to find clusters in lower dimensional subspaces embedded in the original space [128, 160, 167, 222]. Unlike traditional clustering algorithms, in addition to grouping data points into clusters, subspace clustering also needs to simultaneously determine the subspace in which each cluster resides. Figure 1.1 shows an example of subspace clustering results for car parking occupancy of the CBD of the City of Melbourne, which highlights the difference from traditional clustering. The data represents parking occupancy levels at different parking bays (represented by columns) in the CBD at different hours of the day (represented by rows). Here, each parking bay corresponds to a dimension (or feature), and each timestamp corresponds to a data point (or instance). Subspace clustering aims to find not only the grouping of points where values are similar, but also the dimensions in which this similarity is exhibited. A subspace cluster indicates a common pattern of parking availability observed at some parking bays during a certain time of the day. It is likely that each cluster reflects the pattern of small groups of parking bays and is not formed on all dimensions. For example, cluster $C2$ in Figure 1.1 indicates that the car parking occupancies from 11am to 1pm are similar, but only at parking bays $P1$, $P2$, $P3$ and $P4$ but not at all parking bays. Subspace clustering is able to discover these two aspects of each cluster, while traditional clustering algorithms assume the full dimensional space for each cluster. In the next section, we further elaborate on why this assumption makes traditional clustering algorithms ineffective for high dimensional data.

Another application of subspace clustering can be found in the analysis of microarrays of gene expression levels [226], which have become a common source of data in experimental molecular biology. Given the expression levels of tens of thousands of genes collected in various samples (where each sample can correspond to a tissue, experiment, or patient), clustering is commonly used to explore the patterns of expression levels that exist in the data, prior to further analysis. Studies [114, 124, 176] have shown that any pattern is likely to be shared by only some genes (and not all the genes) and observed on only a subset of samples. Therefore, it is critical for the clustering process to be able to identify these two aspects for each cluster. For this reason, subspace clustering has become increasingly popular to analyze gene expression data. This characteristic of subspace clustering has also made it more effective compared to traditional clustering algorithms in other applications such as image segmentation, recommender systems, and text analysis [128, 167].

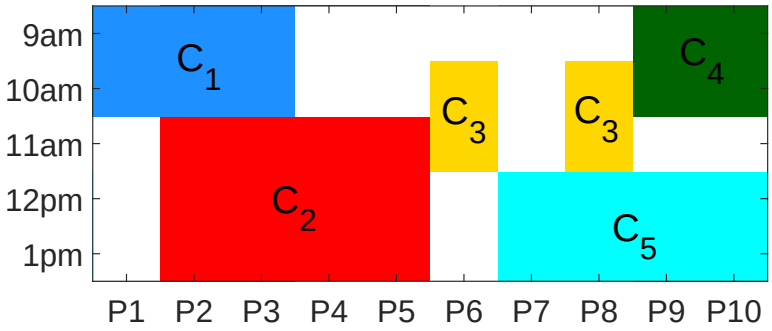


Fig. 1.1 An illustration of clusters in non-disjoint subspaces for car parking occupancy data, where the horizontal axis represents different parking bays, and the vertical axis represents different times of the day. Clusters are highlighted to show simultaneous groupings of points (time periods) and dimensions (parking bays) with similar levels of parking occupancy.

With the rapid development of data intensive applications in the Internet of Things, bioinformatics, and the World Wide Web, there is a strong demand for subspace clustering algorithms that can handle the large volumes of high dimensional data being generated nowadays. However, multiple challenges arise for subspace clustering in such applications due to the difficulty of simultaneously grouping large numbers of data points and the dimensions of each cluster in high dimensional spaces. We introduce these challenges in the next section.

1.1 Challenges for Subspace Clustering

We identify and discuss in detail the challenges that arise when performing subspace clustering high dimensional data. The most fundamental challenge in analyzing high dimensional data is *local feature relevance* [128]. This attribute, in addition to the curse of dimensionality, subsequently raises challenges in measuring similarities between data points in high dimensional space, which is a critical step in any clustering algorithm [110]. Moreover, we identify another open research challenge that is to find clusters in *non-disjoint* subspaces. This challenge arises in multiple applications but is not addressed in many existing algorithms [222, 230].

1.1.1 Local Feature Relevance

Local feature relevance is a challenge that is commonly observed in high dimensional spaces. It means that data points are usually correlated in lower dimensional subspaces, rather than in the full dimensional space. Moreover, subspaces vary for different clusters. This means that the relevance of a feature is only local to a subset of points. In the context of subspace

clustering, this means that different clusters reside in different subspaces. For example, as shown in Figure 1.1, the pattern represented by cluster $C1$ is only applicable to parking bays $P1$, $P2$ and $P3$, while cluster $C2$ is defined in the subspace shared by parking bays $P2$, $P3$, $P4$ and $P5$.

Identifying the relevant subspace for a group of points directly affects how similarities between these points are measured, and therefore is critical to the clustering result. Table 1.1 illustrates this phenomenon and explains why finding local feature relevance is critical to subspace clustering. Given five points $\{x_1, x_2, x_3, x_4, x_5\}$ in a 5-dimensional space, it can be observed from Table 1.1b that the similarities between the points are not revealed if the full dimensional space is considered. In particular, point x_1 is deemed to be more similar to point x_3 than to point x_2 (reflected by the smaller Euclidean distance). In reality, two points x_1, x_2 are identical in the subspace of dimensions $\{d_1, d_2, d_3, d_4\}$. It is the irrelevant dimension (d_5) that distorts the similarity and directly affects the clustering result. This example emphasizes the importance of selectively considering only relevant dimensions when finding subspace clusters.

Table 1.1 The challenge of local feature relevance for traditional similarity measures in multi-dimensional space.

	d_1	d_2	d_3	d_4	d_5
x_1	1	1	1	1	1
x_2	1	1	1	1	10
x_3	4	1	5	1	2
x_4	1	6	3	2	5
x_5	2	1	1	1	7

(a) A set of five points in 5-dimensional space.

	x_1	x_2	x_3	x_4	x_5
x_1	0	9.0	5.1	6.8	6.1
x_2	9.0	0	9.4	7.4	3.2
x_3	5.1	9.4	0	6.9	6.7
x_4	6.8	7.4	6.9	0	5.9
x_5	6.1	3.2	6.7	5.9	0

(b) Euclidean distance between the five points in the full dimensional space.

Previous research has proposed to tackle clustering of multi-dimensional data by first reducing the number of dimensions using dimensionality reduction techniques such as *Principal Component Analysis* (PCA) [117], *Linear Discriminant Analysis* (LDA) [112], *manifold learning techniques* [173], and *feature selection techniques* [43]. However, the existence of different subspaces for different subsets of points makes these techniques inappropriate. All the aforementioned techniques apply *global* changes on all data points, and result in only one lower dimensional subspace, effectively enforcing all points to belong to the same subspace. This procedure contradicts the attribute of *local* feature relevance.

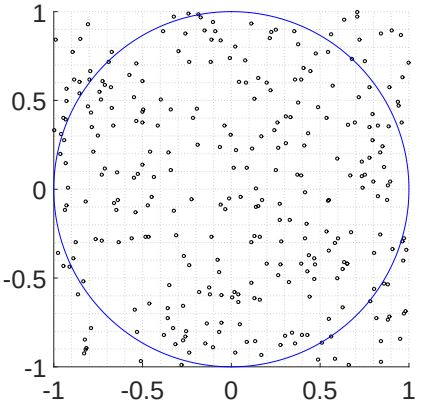
1.1.2 Difficulties of Similarity Measurement in High Dimensional Space

Multiple studies [12, 78, 185] have shown that data become sparse in higher dimensional spaces. As the dimensionality increases, data points are spread further apart and it is exponentially less likely that clusters of points can be formed. Figure 1.2 demonstrates this phenomenon, where data are uniformly distributed in the range $[-1,1]^d$ in a d-dimensional space. In this example, points that are within a distance of 1 from the center are located in the unit hypersphere and can be considered to belong to the same cluster. Therefore, the ratio between the volume of the unit hypersphere and the entire data space (i.e., the volume of the hypercube limited by $[-1,1]^d$) indicates the probability that a point belongs to that cluster.

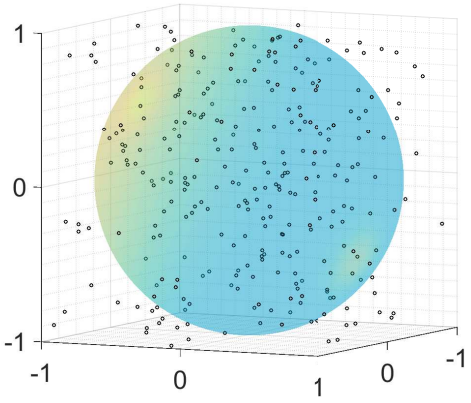
It can be observed that the ratio between the volume of the hypersphere and the hypercube decreases rapidly to 0 as the number of dimensions increases, e.g., the ratio is 2.5×10^{-8} when the dimensionality is 25. Intuitively, given any cluster boundary, Figure 1.2 illustrates the phenomenon that most of the data lies outside of it, and that the probability that points belong to that cluster approaches zero as the dimensionality increases. Studies [12, 231] have shown that the concept of proximity and similarity between points are no longer meaningful and cannot be used in the traditional way for subspace clustering algorithms. Different similarity measurements have been evaluated for high dimensional data, including Euclidean distance (L_2-norm), Manhattan distance (L_1-norm), fractional distance (L_k-norm, $0 < k < 1$), as well as cosine similarity and other distance metrics [12]. However, none of these metrics prove to be effective and stable for high dimensional data. The root cause is that they take all dimensions, both relevant and irrelevant, into consideration when computing the distance. The inclusion of irrelevant dimensions increases the overall distance between two points and spatially spreads them further apart in the full dimensional space. This obfuscates the underlying similarity between points and explains why traditional clustering algorithms do not work well with high dimensional data. It reinforces the importance of selectively retaining only relevant dimensions when computing the similarities for different subsets of points.

1.1.3 Finding Base Clusters in Non-Disjoint Subspaces

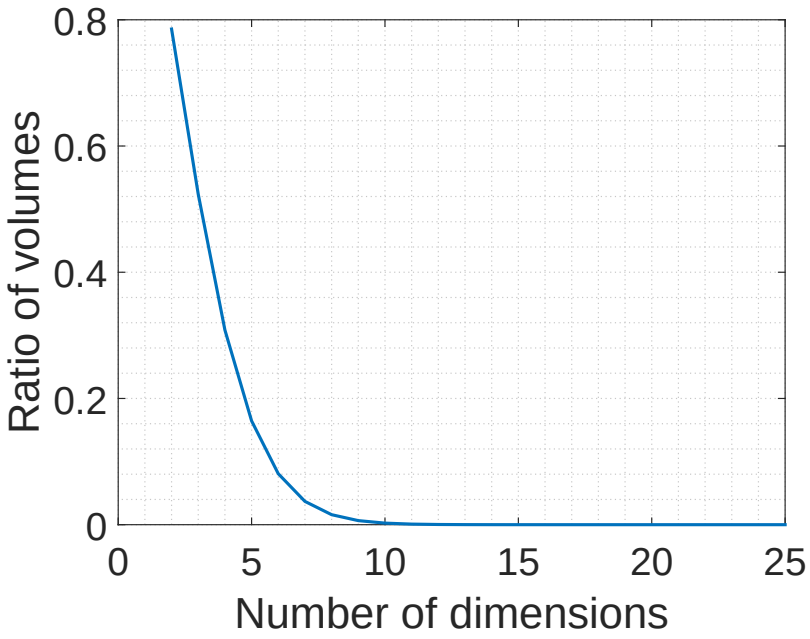
Finding clusters in non-disjoint subspaces remains an open research challenge. Many state-of-the-art algorithms [39, 55, 66, 107, 122, 222] work under the assumption that clusters are located in disjoint subspaces, which do not have any intersection except for the origin. This is a strong assumption that can be unrealistic when clustering high dimensional data, because data are usually correlated in overlapping subsets of dimensions, resulting in clusters forming in non-disjoint subspaces.



(a) $d = 2$



(b) $d = 3$



(c)

Fig. 1.2 (a) The unit circle and unit grid in 2-dimensional space; (b) The unit sphere and unit cube in 3-dimensional space; (c) The ratio between volumes of the unit hypersphere and unit hypercube as the dimensionality increases.

An example of high dimensional data with clusters in non-disjoint subspaces comes from traffic sensors, which is a typical use case of the Internet of Things. When analyzing the traffic volume of a city, traffic counts collected by sensors deployed at each intersection can be considered as a feature (dimension), where a data point is a vector of these sensor measurements giving a snapshot of the traffic volume of the city at a particular time. Clustering this data can provide insights into the correlation of traffic at different locations and times.

A single dimension can commonly be part of multiple clusters, e.g., traffic at an office area can belong to a cluster of congested traffic (e.g., in the mornings), a cluster of moderate traffic (e.g., during noon), and a cluster of very low traffic (e.g., during weekends). Finding non-disjoint clusters of this nature is a challenge for existing subspace clustering methods.

Another example is the analysis of gene expression data, in which a particular gene can be involved in multiple genetic pathways, which can result in different symptoms among different sets of patients [71]. Hence, a gene can belong to different clusters that have some dimensions in common while differing in other dimensions [98].

Clusters in non-disjoint subspaces can also be observed in the previous example in Figure 1.1. Clusters C_1 and C_2 are in non-disjoint subspaces since they share the dimensions of parking bays $P2$ and $P3$ in common. In the case of C_1, this can be interpreted as the utilization of these two parking bays following some pattern that is also observed at $P1$ between 9am-10am. On the other hand, cluster C_2 shows that $P2$ and $P3$ follow a different pattern between 11am-1pm, and share that pattern with $P4$ and $P5$. Further analysis of the data can suggest that $\{P1, P2, P3\}$ are busy parking bays during morning peaks, whereas $\{P2, P3, P4, P5\}$ have higher occupancy levels during lunch time.

The assumption of a disjoint subspace structure of clusters has enabled a simpler formulation of the clustering problem, and has motivated a number of different techniques to be applied in the literature (more details in Section 2). However, the existence of clusters in non-disjoint subspaces in real life scenarios as illustrated above highlights the need for subspace clustering algorithms that relax this assumption.

1.1.4 Scalability to Big Data

With advances in cloud computing technology, the market for big data applications [188, 229] has seen exponential growth in both revenue and public interest. Alongside the opportunities, the development of big data brings novel challenges to many applications. Specifically, the volume of data and the velocity at which data is captured require any analysis technique to be able to scale to very large inputs. In fact, data captured by many applications can grow to millions of records in a short period of time. For example, the City of Melbourne has deployed in-ground sensors at most parking bays in the CBD to capture parking events, and more than 34 million records have been captured for the single year of 2016 [149]. In biomedical studies, the microarrays of gene expression levels often have up to 10,000 dimensions or more.

The ability to scale to large inputs becomes critical for many algorithms to remain practical in the era of big data. It has been shown [160, 222] that many existing algorithms have high computational costs and take considerable time to cluster relatively small inputs,

e.g., STATPC [160], which is an effective subspace clustering algorithm, needs more than 13 hours to cluster 7,500 records of 16 dimensions. This challenge has motivated our focus on developing subspace clustering techniques that are scalable to large datasets.

1.2 Research Focus

Measuring the similarity between points is critical to clustering and directly impacts the result. We experience this challenge in our first motivating experiment of using clustering to model the distribution of pedestrians in the City of Melbourne, which we discuss in detail in Chapter 3. By analyzing only two locations, our clustering-based model is able to find clusters that correspond to different load profiles of pedestrians at different times of the day. This shows that the clustering method works effectively with two-dimensional data in this experiment. However, as the number of dimensions increases to more than two, the clustering results as well as the quality of our model deteriorates rapidly. This illustrates that traditional similarity measures are unable to overcome the challenges of finding local feature relevance and the curse of dimensionality to measure similarities effectively. Therefore, our study begins with designing a new method to measure similarity in high dimensional spaces, and highlights it as a key component of the solution towards subspace clustering.

Q1: *How can we selectively include only relevant dimensions to effectively measure the similarity between data points in high dimensional space?*

Studies [113, 119] have shown that the similarity between points depends on the context, application, and the distributions of the points. We therefore focus on the application of subspace clustering and develop a similarity measure that is context-based, i.e., it is able to rely on data points to selectively include only relevant dimensions when computing the similarity between data points in a high dimensional space.

To this end, we consider similarity in a high dimensional space as the aggregation of similarities in lower dimensional subspaces, which we refer to as the base similarity. These base similarities reaffirm and re-enforce the overall similarity. Our approach is to search for these base similarities in lower dimensional subspaces and subsequently aggregate them together to form the overall similarity.

Next, we address the challenge of finding subspace clusters in non-disjoint subspaces. We focus our study on the strategy of bottom-up clustering. This strategy searches for dense units in each dimension and combines them sequentially to form clusters in the high dimensional space. According to the literature [128, 167, 222], it is the only strategy that implicitly

considers clusters in overlapping subspaces by default. Our similarity measure also follows the same strategy, which allows for easier and more efficient integration.

However, a bottom-up clustering strategy possesses multiple drawbacks, specifically in its reliability for data with complex structures, and its scalability to high dimensional data. Our focus is to not only develop a subspace clustering algorithm using a bottom-up strategy to leverage its benefits, but also to overcome these limitations. In particular, we focus on improving two key components of the bottom-up strategy:

- The existing search for dense units in low dimensional spaces does not preserve the covariances between dimensions. This leads to both (1) valid clusters being missed, and (2) redundant base clusters that exponentially increase the workload of the aggregation phase and consequently reduce the algorithm's scalability.

- The aggregation of the low dimensional dense units is performed sequentially with a complexity that is combinatorial in the number of dimensions. This approach is not only sensitive to noise and outliers, but can also propagate the false aggregation of low dimensional dense units into higher dimensional subspaces.

To this end, our second research question is stated as follows:

Q2: *How can we improve subspace clustering using a bottom-up strategy in terms of (1) its search for low dimensional dense units, and (2) the aggregation of dense units into higher dimensional clusters in order to improve the cluster quality of the subspace clustering algorithm?*

One of the distinct approaches we propose in this thesis that differs from other bottom up subspace clustering algorithms is to start the search for similarities in low p-dimensional base subspaces ($p > 1$) instead of in individual dimensions. This method integrates our proposed similarity measure into the subspace clustering algorithm, which is a non-trivial task, and benefits from its advantages. Subsequently, we propose to use the frequent pattern mining technique to aggregate the lower p-dimensional similarities in order to form higher dimensional clusters. This addresses the second limitation of bottom up clustering algorithms that usually aggregate lower dimensional clusters in a sequential fashion. We elaborate and demonstrate in Chapter 5 that by improving these two key components of bottom up clustering algorithms, our algorithm is more efficient, effective and tolerant to noise and outliers than other subspace clustering algorithms.

Next, we acknowledge that technology keeps evolving and the data generated by real-world applications keep increasing, both in volume and dimensionality. To this end, an important aspect that we focus on in this thesis is the scalability of the algorithm to the

sizes of data that are generated by contemporary real world applications. We state the third research question as follows:

Q3: *How to make our subspace clustering algorithm scale to datasets with large data volumes and high dimensionality?*

We start addressing this research question by first surveying the scalability of existing subspace clustering algorithms, in terms of their capability of handling inputs that have large volumes and high dimensionalities. We also analyze the scales of data that are generated by real world applications in different domains, before stating the expected scope for our research in Section 2.5.

Our consideration of scalability is reflected throughout the proposal of both the similarity measure and the clustering algorithm. We evaluate different approaches to avoid expensive computation and improve the efficiency of the algorithm. In summary, the key approaches that we adopt to improve scalability are:

- We bypass the generation of a similarity matrix, which is a standard process in clustering, in order to reduce the quadratic time and space complexity.
- We use a frequent pattern mining technique to aggregate lower dimensional clusters. This approach is linear to the number of data points and allows the algorithm to scale to inputs that have large volumes.
- The aggregation that uses frequent pattern mining by default removes noisy and irrelevant subspaces (infrequent items) at an early stage and therefore reduces the number of subspaces to be aggregated.
- We identify the low scalability of the proposed algorithm w.r.t the dimensionality, and subsequently propose a subspace sampling technique that allows the algorithm to scale better to larger numbers of dimensions.

1.3 Organization of Thesis and Contributions

This section summarizes the organization and contributions of this thesis, as shown in Figure 1.3. A brief overview of each chapter, along with its main contributions, is presented.

Chapter 2 - Background and Survey of Subspace Clustering Algorithms

This chapter gives an overview of the problem of subspace clustering of high dimensional data. It first presents the basic concepts and challenges of subspace clustering before

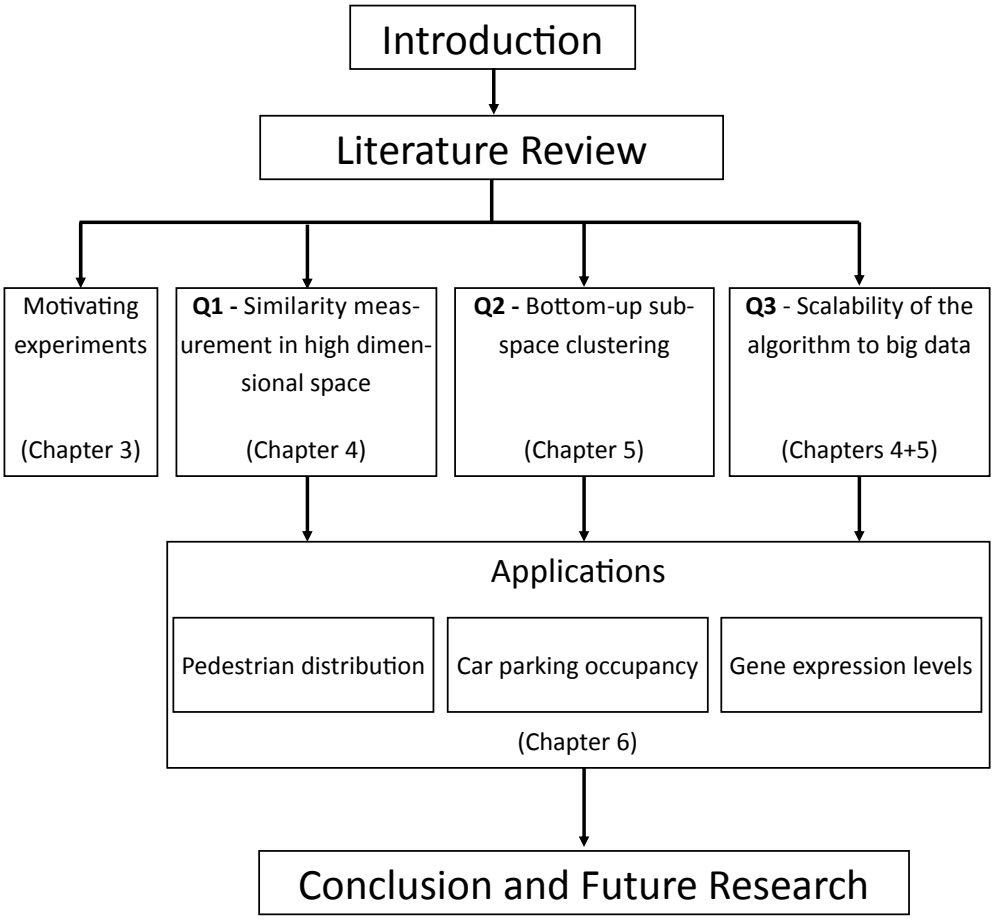


Fig. 1.3 Organization of the thesis.

discussing existing clustering algorithms, their advantages and limitations. We identify the gaps in the existing algorithms and conclude with the research challenges that we aim to address in this thesis.

Chapter 3 - Profiling Pedestrian Distribution and Anomaly Detection in a Dynamic Environment

This chapter presents our first contribution in using clustering to solve a real-life application of the Internet of Things. It serves as motivating work that reveals and illustrates different attributes of high dimensional data as well as challenges in analyzing this type of data.

We model the pedestrian distribution of the City of Melbourne, and use the model to detect anomalous events in a systematic way. We experiment with different techniques for clustering, anomaly detection, and frequent pattern mining to build different models in both

low and higher dimensional spaces (the highest number of dimensions that our experiments are able to handle is only 10 in this chapter). As the number of dimensions increases, we demonstrate the effects of local feature relevance on the algorithms and discuss why it stops them from working effectively. This motivates the importance of overcoming the challenge of local feature relevance to measure the similarity between data points.

Our key contributions in this chapter include:

- We demonstrate that it is feasible to model large-scale urban pedestrian counting data in a dynamic and changing environment. We show how the raw and discrete data of pedestrian counts can be transformed into a meaningful model that describes the patterns of pedestrian activities.
- We demonstrate the challenge of local feature relevance in a real-life application and present initial attempts to tackle this challenge. These experiments emphasize the need for a subspace clustering algorithm to solve this problem more effectively.

The main content of this chapter has been published in **Publication 1** and **Publication 2** listed in the Preface.

Chapter 4 - An Effective Measurement of Similarity in High Dimensional Space

In this chapter, we address a key challenge of subspace clustering, which is to measure the similarity between data points in high dimensional spaces (Question Q1). We propose a novel method that can overcome the challenge of local feature relevance and produce meaningful similarities for the application of subspace clustering.

Our proposed method considers the similarity between any two points as the union of base similarities that are frequently observed in its lower p-dimensional subspaces. To this end, the method searches for similarities in lower dimensional subspaces, and aggregates them to form the overall similarity. We provide proofs of the concept that this approach can be applied for similarity based on distance, density, and grid formulations. This flexibility allows our proposed measurement of similarity to be used in distance-based, density-based, and grid-based clustering.

The novelty of our proposed method is reflected in the following ways:

- The measurement is context-based, in that it considers the covariances between different dimensions and takes into account the values of other data points to derive the similarity. As we further elaborate in Chapter 4, this improves clustering quality and reduces the search space to keep the computational complexity low.

- The proposed method provides a balance between computational complexity and the quality of the measurements. Depending on the available computational resources, attributes of the data, and nature of the application, the proposed method can be easily adjusted by only one parameter to suit the situation.

This similarity measure plays a key role in our subspace clustering algorithm. The initial working version of the method has been published in **Publication 3**, a revised and more thoroughly analyzed version has been included in **Publication 4**.

Chapter 5 - Scalable Bottom-up Subspace Clustering using FP-Trees for High Dimensional Data

This chapter discusses how we integrate our proposed technique on constructing a similarity measure for high dimensional data from similarities in base subspaces in the context of subspace clustering. We first present that our proposed similarity measure, while being able to find point-to-point similarities effectively, cannot be used in its original formulation for a subspace clustering algorithm. In addition to the expensive computation of the similarities, a fundamental challenge remains, which is the context of the subspace in which the similarities are computed. For this reason, it is not valid to perform clustering of points based solely on the numerical values of similarities. Instead, we propose a two-phase algorithm of Subspace Clustering by aggregating Base Similarities (SCBS) that constructs higher dimensional clusters by aggregating similarities in lower dimensional subspaces.

This algorithm first searches for base clusters in low dimensional subspaces. The second phase of the algorithm then forms clusters in higher dimensional subspaces using these base clusters. The aggregation is formulated as a problem of frequent pattern mining [97], which enables efficient search for clusters in higher dimensional subspaces using FP-Trees [36, 97].

We make the following contributions in this chapter:

- We propose novel approaches that are reflected in both phases of the subspace clustering algorithm. First, we show that our proposed similarity measure can be integrated into a subspace clustering algorithm and it helps preserve the covariance of data between different dimensions. Second, we transform the process of sequential aggregation of low dimensional clusters to a problem of frequent pattern mining to construct high dimensional clusters (question Q2).

- We propose a novel subspace clustering algorithm that can find clusters in non-disjoint subspaces and handle inputs having large volumes (up to one million data points). We then analyze its limitation in scaling to high dimensional data, and propose a solution

that allows the algorithm to scale to datasets having 10,000 data points and up to 3,500 dimensions within a reasonable running time (question Q3).

- We demonstrate that the proposed algorithm outperforms traditional subspace clustering algorithms using bottom-up strategies, as well as state-of-the-art algorithms with other clustering strategies, in terms of accuracy and scalability to large volumes of data (question Q3).

The content of this chapter has been published in **Publication 4**.

Chapter 6 - Applications

In this chapter, we evaluate the ability of our proposed subspace clustering algorithm to find clusters in datasets from different real-life applications.

First, we revisit the problem of modeling the pedestrian distribution of the City of Melbourne. We show that our proposed algorithm can overcome the challenges raised in Section 1.2 as well as the limitations of the experiments in Chapter 3. By conducting both qualitative and quantitative evaluation, we demonstrate that SCBS can produce high quality clusters that offer insights on the pedestrian distributions and pedestrian movements at different times of the day. Subsequently, we showcase the capability of the algorithm to analyze the impacts of the Night Network [4], which is a major change in the public transport of Melbourne, on the activities of pedestrians. The clustering results produced by SCBS can reveal both (1) a noticeable change in pedestrian activities at midnights on weekends before and after Night Network, and (2) the locations in the CBD where the change of pedestrian activities is observed. We explain and analyze the clustering results using the technique of iVAT [99, 111] and contrast mining [72], and both indicate consistent findings with the clusters and subspaces that we find.

We then push the algorithm to cluster data that has much higher dimensionalities. Specifically, we use SCBS to cluster ten different gene expression datasets that have up to 10,000 dimensions. In this experiment, we also explore the feasibility of using SCBS as a co-clustering algorithm [31, 174], and benchmark the results with six other well-known co-clustering algorithms. Although SCBS is able to handle the dimensionalities of these datasets, we identify a limitation of our algorithm in clustering sparse data that has high dimensionality but very low volume. We discuss how our algorithm has this limitation and propose the challenge as a future research opportunity to make SCBS work more effectively as a co-clustering algorithm.

Next, we demonstrate that the clustering results of our algorithm can be used as an intermediate step to assist more complicated decision making processes. In particular, we

use the subspace clusters to build an ensemble classification model and show that it improves the accuracy in predicting the car parking occupancy of 276 parking bays in the CBD of the City of Melbourne. This experiment indicates that there is a potential to use our proposed algorithm as an intermediate step to improve other machine learning tasks that analyze high dimensional data.

The applications of the algorithm on gene expression levels and the car parking occupancy has been included in **Publication 4**.

Chapter 7 - Conclusion and Future Research

This chapter concludes by summarizing our research and its contributions. It also provides our reflection on the research journey and discusses future research directions that are motivated by this thesis.

Chapter 2

Background and Survey of Subspace Clustering Algorithms

This chapter provides a review on subspace clustering of high dimensional data. It first presents the basic concepts and challenges of subspace clustering, before discussing existing clustering algorithms, their advantages and limitations. In this chapter we propose a taxonomy of subspace clustering that classifies the algorithms based on the subspace search strategy, as well as the algorithmic approach of the method. As we further elaborate in this chapter, this taxonomy complements the existing surveys of subspace clustering algorithms, which usually focus on only a subset of algorithms we discuss in this chapter and classify them using only either one of the aspects above [128, 167, 222]. In particular, we focus on the ability of algorithms to find clusters in non-disjoint subspaces, their scalability to large input datasets, and the practicality of the assumptions they make. The chapter concludes with the research challenges that we aim to address in this thesis.

2.1 Definitions

Clustering is a point grouping task, where data points in the same group (i.e., *cluster*) are similar to each other, while data points in different clusters are dissimilar [236]. The similarities between data points are measured using a given similarity metric.

Clustering was first proposed in 1938 [32] and has since been used in a wide range of applications including marketing, astronomy, and geology. Recent growth of data volumes has resulted in large datasets with high dimensionality. This raises new challenges that make traditional clustering algorithms ineffective and unable to find correct and meaningful clusters [64, 128, 167, 189]. These challenges require adaptations of clustering algorithms to cope with the characteristics of high dimensional data, in order to maintain cluster quality

while managing the computational complexity. This leads to the proliferation of subspace clustering algorithms.

Subspace clustering aims to find clusters in lower dimensional subspaces embedded in the original space. Unlike traditional clustering algorithms, besides grouping data points into clusters, subspace clustering also needs to simultaneously determine the subspace in which each cluster resides. Specifically, let:

- $S_s^{(k)}$ be an axis-parallel subspace of k dimensions, which is represented as a set of its component dimensions: $S_s^{(k)} = \{d_{s1}, ..., d_{sj}, ..., d_{sk}\}$, where d_{sj} represents the j^{th} dimension of subspace S_s.
- X_j or $\{x_j\}$ be a set of points; $x_j = \{x_{jp}\}_{p=1}^{k}$ denotes a point, where x_{jp} is the coordinate in the p^{th} dimension of x_j.
- $C_{S_i}^{X_j}$ be a cluster formed by point set X_j in subspace S_i.

The problem of subspace clustering can be defined as follows: Given a set of points X that exist in a d-dimensional space, i.e., $X = \{x_i \in \mathbb{R}^d : i = 1..n\}$, subspace clustering aims to find the set of clusters $Y = \{C_{S_s}^{X_j},\ s : 1..u,\ j : 1..c\}$ that exist in unknown lower subspaces. Here u denotes the number of subspaces, each of which accommodates at least one cluster, and c denotes the number of all clusters. Note that more than one cluster can exist in a subspace, i.e., $c \geq u$. *In this thesis, we consider subspace clustering as the problem of identifying the corresponding subspaces and data points of all clusters.*

Subspace clustering needs to simultaneously determine the subspaces in which a cluster can potentially exist and the partitions of the points in each subspace. This additional requirement brings multiple challenges for subspace clustering as discussed in detail in the next section.

2.2 Challenges

In this section we discuss the fundamental challenges that subspace clustering needs to address. The challenges also explain why traditional clustering algorithms are unable to work effectively with high dimensional data.

2.2.1 Local Feature Relevance

One of the fundamental challenges in analyzing high dimensional data is *local feature relevance* [128], which states that data points are usually correlated in subspaces, rather than in the full dimensional space. Moreover, subspaces vary for different subsets of data.

This phenomenon has been observed throughout different tasks of analyzing high dimensional data. Tzikas et al. [218] propose a Bayesian learning algorithm over high dimensional data. They observe that feature selection is local, and different values apply to different kernels of the algorithm. Therefore, there is no single set of features that are significant to all instances of the input. Instead, different features are considered significant for different subsets of the input space. This study re-iterates the property that the relevance between features is only local in high dimensional spaces.

Melo et al. [151] compare the performance of global and local feature selection in the classification of high dimensional data. They consider datasets that have various flat and hierarchical structures collected from different application domains. Their empirical results show that the approach using local feature selection consistently outperforms the global approach in terms of predictive performance. This work demonstrates that relevance is only local to a subset of features, and that features are relevant in different ways for different subsets of points.

Armanfard et al. [19] highlight the importance of local feature relevance as opposed to typical global feature selection in data classification. Each subset of data points therefore is associated with its own distinct optimized features, which may vary both in membership and size across different sets. The authors propose a novel approach to find the optimized set of relevant features for each data subset. They demonstrate that localized feature selection produces better results than a global feature selection method.

Most of the aforementioned studies discuss and address local feature relevance in the problem of supervised learning. The challenge is also discussed in the context of clustering high dimensional data, albeit not equally popular.

Freitas et al. [84] show the importance of local feature relevance in the application of image retrieval. Their study relies on the hypothesis that similar images can be clustered in some feature subspaces. They use local feature extraction and relevance feedback to cluster similar images in order to narrow down the search space of an image query. The proposed approach is evaluated using more than 30,000 images and shows good performance.

The concept of local feature relevance for high dimensional data in clustering is detailed in [128, 167]. Studies have proposed to address this challenge by reducing the number of dimensions using dimensionality reduction techniques such as Principal Component Analysis (PCA) [117], Multidimensional Scaling (MDS) [35], and feature selection methods [43]. PCA identifies the important dimensions and summarizes d-dimensional data using only k principal components ($k < d$), effectively reducing the number of dimensions from d to k. Likewise, MDS can be used to preserve the similarity between data points while presenting the data in a lower dimensional space. However, these techniques apply global changes to

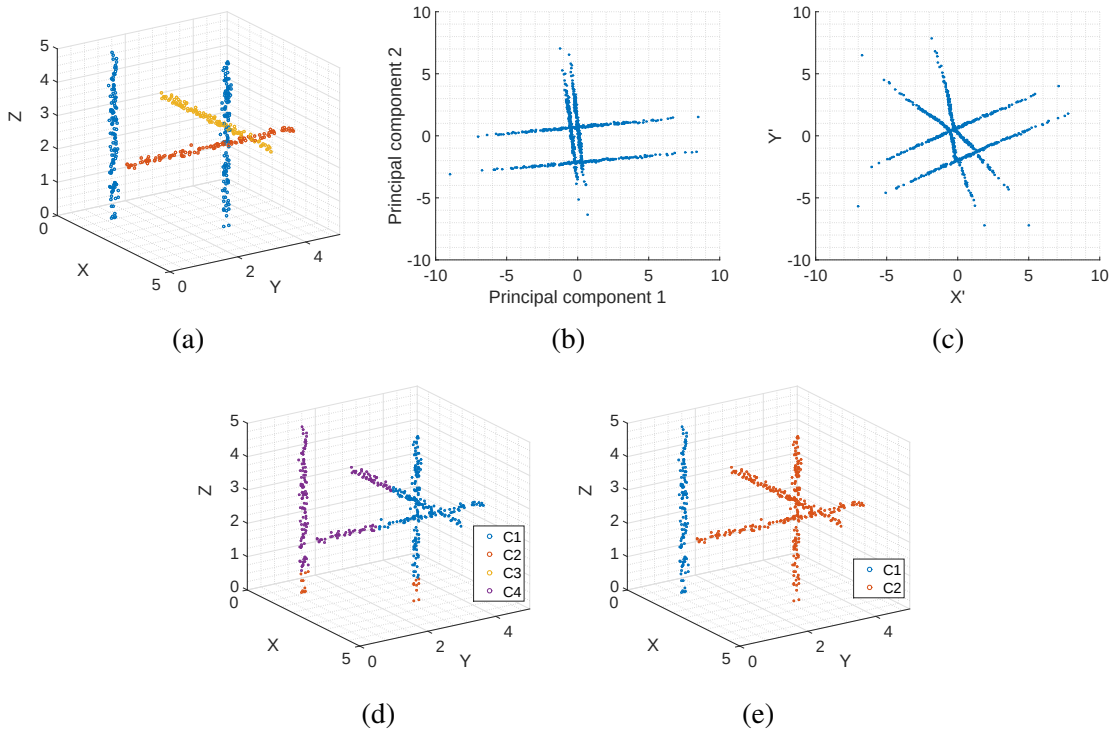
the data and cannot be used as a pre-processing step for clustering high dimensional data. Note that the property of local feature relevance also states that the subspace where a cluster exists varies for different (and unknown) subsets. All the aforementioned techniques result in only one final lower dimensional subspace, effectively enforcing all points to belong to the same subspace, which is a false assumption in general.

Figure 2.1 presents a simple example of a 3-dimensional dataset to further illustrate the property of local feature relevance in high dimensional data. The dataset consists of 800 points in a 3-dimensional space as shown in Figure 2.1a. We use colors to highlight the different subspaces in which some subsets of the data points are correlated and can form clusters. Specifically, the blue points are densely located in the subspace constituted by dimensions $\{x, y\}$ and can form two clusters. The red points form one cluster in subspace $\{x, z\}$ and the yellow points form one cluster in subspace $\{y, z\}$. Note that this information of the ground truth labels of the subspaces is not available in advance in practice. Figures 2.1b and 2.1c show the output of PCA and MDS, which reduce the data to a 2-dimensional space. Although PCA and MDS are able to reconstruct the structure of the data (4 visually separated blocks), the data points are not as well separated as before, as irrelevant dimensions are also taken into account when distances are computed. Therefore, they distort the relative distances between points. Nevertheless, it is harder to find clusters. Note that although it does not seem to be a major issue in this example, the challenges become much more severe as the number of dimensions grows. Both kmeans and DBSCAN fail to find correct clusters, as shown in Figures 2.1d and 2.1e. The inaccurate result suffers from the same challenge as before: irrelevant dimensions make the Euclidean distance less meaningful in expressing the similarity between points.

For this reason, taking all dimensions into account when computing similarity will include both relevant and irrelevant dimensions and therefore lead to poor clustering accuracy [40, 64, 128]. In addition, as the dimensionality grows, it is observed that the concept of distance and proximity between points becomes less meaningful and also affects clustering results.

2.2.2 Curse of Dimensionality

It has been shown that data points are spread further apart as the number of dimensions increases, to an extent where they are considered equidistant from each other in very high dimensional spaces [12]. In this study, the authors show that the difference between the maximum and minimum distance between points is independent of the data distribution and do not provide any intuition about proximity between points in high dimensional space. Specifically, for an arbitrary distribution of n points $\{x_i\}_{i=1}^{n}$ in d-dimensional space, Aggarwal



(a) (b) (c)

(d) (e)

Fig. 2.1 (a) An example 3-dimensional dataset; (b) 2-component PCA of the data; (c) Multidimensional scaling plot of the data; (d) kmeans clustering result; (e) DBSCAN clustering result.

et al. [12] show that:

$$C \leq \lim_{d \to \infty} \left[\frac{Dmax_d^k - Dmin_d^k}{d^{1/k-1/2}} \right] \leq (n-1)C$$

where C is a constant, *Dmax* and *Dmin* are the maximum and minimum distance between any points, k indicates the order of the norm, e.g., L_1 norm is Manhattan distance, L_2 norm is Euclidean distance. This result shows that the Manhattan distance ($k = 1$) between any two points diverges to ∞, and the distance using other metrics ($k > 1$) is bounded by a constant C_k (Euclidean distance converges to 0, i.e., $C_2 = 0$). Therefore, they do not offer any useful information about the similarity between points in high dimensional space.

This behaviour of high dimensional data emphasizes the importance of selecting only the relevant dimensions while computing similarity, and further explains the reason why traditional clustering algorithms do not work with high dimensional data.

2.3 Evaluation Metrics

Validation of clustering results poses multiple research challenges. Due to the lack of ground truth labels, clustering performance is usually assessed by internal evaluation, where the memberships of points to clusters are transformed into a single clustering quality score. Cluster quality can also be evaluated using external measures when ground truth labels (or a gold standard) exist. In this section, we present different methods to perform internal and external evaluation of clustering results.

2.3.1 Internal Evaluation

Internal evaluation is used when there is a lack of ground truth labels. Internal evaluation methods are based on the intuition that data points in the same cluster are more similar than data points in other clusters [183]. Therefore, these evaluation methods assign high scores to clustering results that have high intra-cluster similarity and low inter-cluster similarity. There are multiple internal evaluation measures [79, 171, 183]. We discuss three popular ones below.

Davies-Bouldin (DB) Index [171] is computed as the sum of the ratio between intra-cluster and inter-cluster distances:

$$DB = \frac{1}{n} \sum_{i,j=1}^{n} \max_{i \neq j} \frac{\sigma_i + \sigma_j}{d(i,j)} \tag{2.1}$$

where σ_i, σ_j are the average intra cluster distances of clusters c_i and c_j, and $d(i,j)$ is the distance between centers of clusters c_i and c_j. A good clustering result has low intra-cluster distances (high intra-cluster similarity) and high inter-cluster distance, hence a low DB index.

Dunn (D) Index [79] targets the quality of separating data into well separated dense areas, specifically:

$$D = \frac{\min_{i \neq j} d(i,j)}{\max \sigma(k)} \tag{2.2}$$

The numerator is the minimum distance between any two cluster centers c_i and c_j (the worst case of inter-cluster distances), and the denominator represents the worst case of intra-cluster distances among all clusters. The index therefore measures the worst-case separation of clusters. Hence, a good clustering result should be able to separate the clusters and have a high Dunn Index.

Silhouette (S) [171] takes into account both the cohesion of each cluster and the separation between different clusters. This index measures clustering quality on a data point level. Specifically, the quality of an assignment of point x to cluster c is measured by:

$$s(x) = \frac{d(x) - \sigma(x)}{\max(d(x), \sigma(x))} \tag{2.3}$$

where $d(x)$ is the average distance between point x and all points of other clusters, so the higher $d(x)$ is, the better is cluster separation. $\sigma(x)$ is the average intra-cluster distance between x and other points of the same cluster. Using Equation 2.3, the quality of assigning x to a specific cluster is in the range $[-1, 1]$. A value near 1 indicates good clustering quality, while a value near -1 indicates poor cluster separation and the point should belong to another cluster. Silhouette index can also prevent cases where only one cluster is produced (that fits the entire data and does not offer new knowledge), or there are too many clusters. The overall clustering quality is computed by taking the average of the silhouette indices of all points. It has been shown that Silhouette can also be used to determine the number of clusters within a dataset [60].

One of the drawbacks of internal evaluation measures is that they are unable to reflect the validity of clusters [79], i.e., a high score does not always translate to the effectiveness and usefulness of the result. In addition, each index works on a certain assumption about the underlying structures that exist in the data and therefore is only valid if this assumption holds. For example, the aforementioned measures assume convex clusters [30], and in fact favour spherical clusters.

2.3.2 External Evaluation

External evaluation refers to evaluations that use additional information not used for clustering to evaluate the effectiveness of the clustering algorithms. This information is usually referred to as a ground truth label, or gold standard. Even with the existence of ground truth labels, evaluation of clustering results is more challenging than verification of classification, which is simply a problem of exact value matching of the predicted label and the actual class [175]. The challenges are caused by the permutation of cluster labels. For example, an algorithm can partition 5 points $\{x_1, x_2, x_3, x_4, x_5\}$ into 2 clusters such that the cluster assignment is as follows $(c_1, c_1, c_1, c_2, c_2)$, i.e., the first three points are grouped into cluster c_1 while the last two points are grouped into cluster c_2. This result is equivalent to another clustering result of $(c_2, c_2, c_2, c_1, c_1)$, because clustering aims to discover the relative grouping of similar data points together, rather than the exact assignment of cluster labels.

Nonetheless, most existing measures of external evaluation are adapted from classification-based evaluation criteria [172, 146]. To this end, a *true positive* is defined as an instance of two points having the same class and being grouped into the same cluster. Similarly, the number of true negatives are counted as the number of times two points having different class labels belong to two different clusters [146]. Having the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), different measures can be computed as follows.

$$RandIndex = \frac{TP+TN}{TP+FP+TN+FN} \tag{2.4}$$

Rand index (RI) measures how similar the clustering result is, as compared to the ground truth label [204]. Intuitively, it is the percentage of the number of correct clustering decisions. A drawback of RI is that it weights false positives and false negatives equally, which may not be desirable in many applications, such as medical applications, where false negatives should be penalized more than false positives. This problem is addressed by the chance-corrected adjusted rand index [204].

Clustering results can also be evaluated by *precision*, *recall*, and *F-measure* [175].

$$Precision = \frac{TP}{TP+FP} \tag{2.5}$$

$$Recall = \frac{TP}{TP+FN} \tag{2.6}$$

$$F_\beta = \frac{(\beta^2+1) \times Precision \times Recall}{\beta^2 \times Precision + Recall} \tag{2.7}$$

where β can be used to adjust the weight of *Recall* in $F-measure$ depending on the requirements of the application. Other classification-based external evaluation measures include Jaccard Index, Dice Index, and Fowlkes–Mallows Index [157].

Another approach to benchmark clustering results with the ground truth labels is to use mutual information [172]. This index reflects the amount of information shared between the two sets of labels, i.e., a high value indicates a good clustering result. Specifically, let P be the cluster labels predicted by a clustering algorithm, and C be the true classes, the normalized mutual information (NMI) [224] is defined by:

$$NMI = \frac{H(P)-H(P|C)}{\sqrt{H(P) \times H(C)}} \tag{2.8}$$

where $H(\bullet)$ denotes the entropy; the numerator is the mutual information and the denominator normalizes the index to the range of $[0, 1]$.

2.4 Taxonomy of Subspace Clustering Algorithms

Subspace clustering consists of discovering two aspects: the subspaces in which the clusters exist, and the groupings of points in each subspace [128, 167]. Different algorithms have different strategies as well as priorities towards finding these two aspects. For example, some algorithms aim only to identify the partitions of data points while ignoring the subspace of each cluster. Such algorithms target applications where knowing the subspace of the clusters offers no useful information. Some of these applications include computer vision and image recognition [76, 135, 242], where it is more important to detect different objects in the image. On the other hand, for other applications, it is important to know both the partitions of data points as well as the dimensions based on which such partitions are made. One such application is clustering of gene expression data [50, 71], where each data point represents a gene and each dimension represents a sample (a tissue or experimental condition). In this application, it is important to reveal both the groups of genes that share common expressions as well as the samples in which these expressions are recorded. The above differences lead to multiple taxonomies of subspace clustering algorithms [167, 128, 222]. We discuss two classification schemes in this section.

The first classification scheme focuses on the strategy to search for subspaces of clusters. The full search space consists of $2^d - 1$ possible subspaces for a d-dimensional dataset and it is computationally infeasible to traverse all these subspaces. How to construct and navigate through the search space plays an important role in the efficiency and effectiveness of any subspace clustering algorithm. Based on the subspace searching strategy, subspace clustering algorithms are divided into two groups [128, 160, 167]: *bottom-up approaches* and *top-down approaches*.

Focusing solely on the search strategy for subspaces can potentially undermine the substantial features of the algorithms. In fact, in some cases the underlying techniques to find subspace clusters are more sophisticated and reflect better the characteristics of the algorithms. Therefore, another popular taxonomy of subspace clustering is to classify the algorithms according to their algorithmic approaches [217, 222, 249]. Under this classification scheme, subspace clustering algorithms are classified into five groups: iterative methods, algebraic methods, statistical methods, matrix factorization-base methods, and spectral clustering based methods.

2.4.1 A Search Strategy-based Categorization of Subspace Clustering Algorithms

Bottom-up Clustering Algorithms

Bottom-up algorithms overcome the challenge of local feature relevance by not directly computing the similarity between points using all dimensions. Instead, such algorithms identify dense units in each dimension and subsequently aggregate those units to form multidimensional clusters. To this end, through the aggregation process, only selected dense units in relevant dimensions contribute to the corresponding similarity between two points. These algorithms then use the downward closure property [238] of density to reduce the search space. In general, if the objects are located densely in subspace S_i, they are also dense in any subspace $S_i' \subset S_i$. If the objects are not dense in subspace S_i', i.e., no cluster exists in S_i', then it is not necessary to search for clusters in all superspaces $S \supset S_i'$. With this intuition, the algorithms start by determining the dense units in each dimension, then subsequently build the k-dimension dense units by combining the $(k-1)$-dimension dense units until the clusters cannot expand to new subspaces with higher dimensions. Since the approaches to aggregate these dense units are not significantly different, the main difference that characterizes each bottom-up algorithm lies in its definition of dense units. The existing algorithms can be categorized into two groups, which are cell-based approaches, and density-based approaches [167].

Cell-based approaches. CLIQUE [14] was one of the first subspace clustering algorithms. CLIQUE divides data points in each dimension into fixed-size cells using their coverage percentage. Using a threshold of coverage, the algorithm discards the cells that cover smaller portions of points. The retained bins of densely populated points are subsequently aggregated to form higher dimensional clusters. The aggregation uses a greedy growth strategy to aggregate adjacent dense units of each dimension in a sequential fashion. Specifically, 1-dimensional dense units are combined to form 2-dimensional clusters. In general, adjacent $(k-1)$-dimensional clusters are combined to form k-dimensional clusters until the cluster can no longer be expanded, i.e., when there is no further adjacent k-dimensional cluster. This process results in a large number of overlapping clusters. Next, the algorithm prunes the redundant clusters by removing small clusters, which are already covered by larger clusters in higher dimensional subspaces. One significant advantage of CLIQUE is that it can find clusters in overlapping subspaces. In addition, since each cluster is constructed by combining dense units, clusters can be presented in an interpretable way and take any arbitrary shape. This is in contrast to other subspace clustering algorithms that favour spherical clusters [236] as presented in the next sections. One of the limitations of CLIQUE is the intuitive parameter

configuration. The algorithm requires the tuning of the grid size in each dimension as well as the coverage threshold, which is hard to know in advance. Further, the value ranges and data density differ in different dimensions; there might not be a single set of parameters that suit all dimensions. In addition, CLIQUE does not scale well with the number of dimensions. If there are on average m dense units in each dimension, the first level of aggregation (to combine 1-dimensional dense units to 2-dimensional clusters) would need to check m^d pairwise possible aggregations, where d is the number of dimensions of the input.

ENCLUS [51] follows the same framework of CLIQUE but uses entropy-based metrics to find dense units and construct subspace clusters. Low entropy indicates high density and high coverage of data cells in each dimension. The algorithm uses the concept of *interest* to measure the correlation between dimensions in a subspace. Interest is defined as the difference between the sum of the entropy of the data in each dimension and the entropy of the data in the subspace constituted from those dimensions [51]. The algorithm then uses these two metrics in the aggregation phase to prune irrelevant subspaces in an Apriori [238] fashion in order to reduce the search space. Specifically, a qualified subspace needs to have a high entropy (dense distribution) and high interest (strong correlation between dimensions). All unqualified subspaces are pruned. ENCLUS benefits from the ability to find clusters in overlapping subspaces with arbitrary shapes. It is also scalable to inputs having large volumes but not to the dimensionality of subspaces. ENCLUS shares the same limitation of CLIQUE in searching for dense units in a static grid: having a single grid size across all dimensions is an impractical assumption and could prevent the algorithm from working with real-life data.

MAFIA [88] overcomes the above limitation by utilizing a histogram to compute grid sizes adaptively to the data distribution in each dimension. It then proceeds like CLIQUE to construct higher dimensional clusters. The algorithm features parallelisation and therefore runs much faster than CLIQUE. Empirical results have shown that it can cluster large inputs with high dimensionality, but the scalability is still constrained by the number of dimensions of clusters [167, 128].

By using simple metrics to find dense units, the aforementioned algorithms possess low time complexity in terms of the volume of data. However, these simple techniques do not work well with noisy data or data with high levels of outliers. CLTree [138] uses a more sophisticated technique to partition each dimension into dense and sparse areas. It inserts hypothetical uniformly distributed data objects and applies a decision tree [29] algorithm on this updated dataset to classify it into different ranges on each dimension. Eventually, traversing through the decision tree and checking the branches that contain the real data provides the dense cells (the tree branches that contain mainly hypothetical data imply sparse

cells of the original data). Although CLTree handles noisy data better, the construction of a decision tree in every dimension is time consuming, especially for numerical data, for which the algorithm needs to explore different split points for the best splitting of branches.

Density-based approaches. Density-based approaches rely on the density of each data point in lower dimensions to form clusters in subspaces. The density of each data point is determined by the number of points within its ε-radius neighbourhood. Although Euclidean distance is usually used to compute this density, it is initially applied only in low dimensional subspaces. These dense areas are then aggregated in a bottom-up fashion. Therefore density-based approaches can discard irrelevant dimensions and ensure that the measurement of similarity only takes into account meaningful dimensions, i.e., dense areas in irrelevant dimensions are not aggregated to form higher dimensional clusters.

SUBCLU [120, 248] uses DBSCAN [77, 129] to search for dense units in each dimension, and uses the downward closure property of density to construct higher dimensional clusters. This means that if a subspace does not contain any cluster (dense area), it is not necessary to check any of its super subspaces. Both DBSCAN and SUBCLU have the same definition of the density of data points, the latter however can find subspace clusters more accurately because it can restrict the distance and density computation to only the relevant dimensions. This requires SUBCLU to invoke a call to DBSCAN for each candidate subspace, which leads to higher computational complexity and running time.

A limitation of SUBCLU is to have a single density threshold for subspaces of different dimensionalities. Multiple studies [12, 234] have shown that this assumption does not hold because points tend to be stretched further apart from each other as the number of dimensions increases, resulting in decreasing density. For this reason, using a single density threshold will not work effectively if the dimensionality becomes too large. A high threshold on density is ideal in low dimensional subspaces but will hinder the formation of high dimensional clusters. On the other hand, a lower density threshold enables the detection of high dimensional clusters but can potentially mistakenly include noise and outliers into low dimensional clusters, or combine these small valid clusters into big clusters. DUSC [21] overcomes this challenge by using varying density thresholds adaptive to the number of dimensions of the subspaces. However, the varying density invalidates the downward closure property of density. Therefore, subspace pruning to reduce search space is not allowed.

Vahdat et al. [219] apply the x-means clustering technique on each dimension to detect the dense cells, and use multi-objective genetic algorithms [61] in two stages. The first stage grows the candidate subspace clusters (CSC). The objective function of the algorithm in this stage is to maximize the intra cluster compactness. The second stage merges the adjacent clusters to yield the optimal combination of CSVs. The objective function of this stage

aims to simultaneously maximise the intra-cluster compactness and minimize the cluster connectivity. Minimizing cluster connectivity ensures only a small portion of neighbouring points are allocated to different subspace clusters, i.e., a dense area is not split into multiple clusters.

Top-down Clustering Algorithms

Top-down clustering algorithms start with an initial approximation of the clusters in the full dimensional space, where all dimensions are assumed to be relevant and assigned equal weights. The algorithms subsequently refine the subspaces through multiple iterations by adjusting the weights of each dimension toward the subspace. Effectively, top-down approaches overcome the challenge of local feature relevance by assigning low (or zero) weights to irrelevant dimensions and high weights to dimensions in which a subset of data points is highly correlated and can form clusters. The weights are updated with the objective function of improving cluster quality in that subspace, e.g., reducing the weights of dimensions in which the intra-cluster distances are large. These algorithms perform clustering in each subspace, resulting in high computational complexity. For this reason, one feature commonly used in these algorithms is to perform sampling on data points in order to improve performance.

We present the traditional algorithms PROCLUS [13] and FINDIT [232], and the more recent algorithms including PROFIT [182] and MAFCA-S [181] that use the top-down approach.

PROCLUS was the first top-down subspace clustering algorithm [128]. It requires the number of initial clusters k and the average number of cluster dimensions l as input parameters. The algorithm first samples the data points and then performs a random selection of k' potential cluster centers C ($k' > k$). k centers are then selected from C as the current cluster centers. Subsequently, the distances of all points in each cluster to its center are computed in each dimension. With the objective of increasing cluster quality, it keeps only l dimensions in which the average distances are the smallest and discards the irrelevant dimensions where the distances are large. This increases cluster compactness. Each iteration is repeated with a different set of cluster centers picked from C. As long as the cluster quality increases (i.e., the average distances in the final l selected dimensions decrease), these new cluster centers are better and replace the previous ones.

FINDIT contains three phases, namely a *sampling phase*, *cluster forming phase*, and *data assignment phase*. The sampling phase produces two data sets from the original data. The first set S is a smaller sampled dataset that the algorithm can work with more efficiently. The second set M contains the potential medoids of the clusters in the original dataset. In the

second phase, the algorithm determines the relevant dimensions for each cluster using its proposed *dimension-oriented distance* (*dod*). This distance requires a threshold ε and states that the relevant dimensions between two points are the ones in which the 1-dimensional distance is within ε. By computing the *dod* between the medoids in M to the points in S, the algorithm votes for the popular dimensions and determines the relevant dimensions for each cluster. The medoids that are near to each other are subsequently merged. This process is repeated multiple times with enlarging ε and eventually an evaluation criterion is used to assess the soundness and finalise the value of ε. The output of the *cluster forming phase* forms the skeleton of the subspace clusters in the original dataset: The group of medoids indicate where the clusters are and the voted dimensions indicate which dimensions are relevant. The last phase assigns every point in the original data to the nearest medoid (distances are computed in the relevant subspace only). FINDIT has great scalability to the volumes of inputs. Woo et al. [232] demonstrated that it can cluster a 30-dimensional dataset that has 5 million data points in 200 seconds. Note that FINDIT performs sampling of data points in this experiment, and the second phase of the algorithm finds clusters on only 250,000 points (5% sampling rate). The remaining points are assigned to the nearest discovered cluster by comparing the distances to clusters. While this sampling technique with low rate can be criticized as discarding potential clusters in the full dataset, FINDIT shows that by satisfying the Chernoff bounds [45], the sampled dataset can guarantee to preserve the cluster structure of the full dataset. This approach motivates a subspace sampling technique that we propose in Chapter 5 to improve the time complexity of our algorithm while still guaranteeing that all dimensions are equally represented in the sampled subspaces.

Recent algorithms have been inspired by the same framework but adopt more advanced techniques. PROFIT uses PCA to reduce the data to its first principal component before performing data sampling. It then divides the data into k partitions before computing the trimmed mean [205] of each partition, which serves as the cluster center of that partition. Subsequently, the Fisher score [91] is used to select the relevant dimensions that minimize the intra-cluster distances and maximise the inter-cluster distances. MAFCA-S is similar to PROFIT but uses the statistical mode of the partitions to allocate the cluster centers. It then uses Median Absolute Deviation (MAD) [133] to measure the dispersion of the data in each dimension, which is subsequently used to refine the relevant dimensions of each cluster. Specifically, only l dimensions (l is provided as the average dimensionality of each subspace that contains clusters) with lowest dispersion are retained. Experiments show that these algorithms are able to handle noisy data with complex structure.

In summary, top-down clustering algorithms require heavy computation in each iteration and therefore require sampling as an intermediate step to improve efficiency. They also

suffer from similar limitations. First, the number of clusters as well as the dimensionality of each cluster are two critical input parameters that affect the results. The former is used for initialization and the latter is required for the cluster and subspace refinement process. However, anticipating these values in advance is challenging. In addition, the number of dimensions is invariant for all clusters, which arguably is an unrealistic assumption and might hinder the detection of clusters with different dimensions, which is a commonly observed phenomenon in real-life data [128, 160, 167]. Finally, the algorithms of this group favour hyper-spherical clusters and are not able to detect arbitrarily-shaped clusters.

2.4.2 An Algorithmic Categorization of Subspace Clustering Algorithms

The previous section describes the categorization of algorithms based on their search strategy for subspaces. Sometimes the segmentation of points and subspaces are not well separated [222]. One such example is the case in which subspace clustering is achieved by solving a joint optimization problem, where both the selection of the points and dimensions of clusters are simultaneously refined [203, 228]. Following such an approach does not explicitly reveal the subspace search strategy. In addition, focusing only on the search strategy for subspaces oversimplifies the overall design of the techniques that define the algorithms.

In this section we discuss more recent algorithms, in terms of the dominant factors that characterize their methods and algorithmic approaches. To this end, subspace clustering algorithms can be divided into five groups: iterative methods [11, 244], matrix factorization based methods [55, 39, 207], polynomial algebraic methods [223], statistical methods [213, 66], and spectral clustering based methods [217, 222, 249].

Iterative methods

The intuition behind iterative methods is as follows: if the segmentation of the data points is known, the subspace of each cluster can be determined using PCA to derive the principal components of each cluster; on the other hand, if the subspaces are known, finding clusters in each subspace can be solved by traditional clustering algorithms. To this end, iterative methods alternate between the segmentation of data points and the refinement of subspaces and repeat these two steps until convergence, i.e., until there is no major change in the evaluation criteria, to improve cluster quality [76, 222].

A representative algorithm of this group is K-subspaces [11]. The algorithm needs the number of subspaces k and the dimensionality of each subspace, i.e., $\{|S_i|\}_{i=1}^{k}$. The objective of the algorithm is to minimize the sum of the Euclidean distances between each data point and the cluster center in its subspace. The algorithm starts with an initialization of the

subspace and point segmentation, and subsequently alternates between refining the points and dimensions of each cluster. In each iteration, the algorithm fixes the subspaces and distributes the points into each subspace so that the sum of distances between points and their corresponding cluster center is minimized. Next, the algorithm performs PCA on each of those clusters (in the full dimensional space) to refine the subspace. This process is repeated until convergence. An advantage of this algorithm is its simplicity, and its ability to find clusters in overlapping subspaces. In addition, the algorithm is guaranteed to converge, albeit only to a local optimum. In practice, multiple iterations of the algorithm with different initializations might allow it to converge to a global optimum. A limitation of the algorithm is the model selection: it not only requires the number of subspaces to be given, but also the dimensionality of each subspace, which can be difficult to obtain in practice. A variant of K-subspace is K-flats [244], which uses Manhattan distance instead of Euclidean distance, in order to reduce the effect of outliers.

Matrix factorization based methods

Matrix factorization based methods assume that clusters reside in independent subspaces, and factorise the input to reveal the affinity matrix [222]. The intuition of these algorithms is that by reordering the rows and columns of the input, cluster structure will be revealed. Therefore, they seek to find this permutation matrix. With the assumption that the subspaces are independent, it is possible to express the permuted input as:

$$X_{permuted} = XV = [U_1, U_2, .., U_k] \begin{bmatrix} \Sigma_1 & & & \\ & \Sigma_2 & & \\ & & \ddots & \\ & & & \Sigma_k \end{bmatrix} \tag{2.9}$$

where U_i is the basis of the subspace and Σ_i is the low dimensional projection of the points in the subspace S_i, i.e., $U_i\Sigma_i$ represents the projection of the points in subspace S_i. Equation 2.9 can be rewritten as:

$$X = U\Sigma V^{-1} \tag{2.10}$$

To this end, Costeira et al. [55] find V by finding the SVD of X, while Boult et al. [39] performs SVD on the row echelon canonical form [207] of X. Kanatani et al. [122] have

shown that the affinity matrix Z can be computed by $Z = VV^T$, where $Z_{ij} = 0$ if the points x_i, x_j are in different subspaces, and $z_{ij} = 1$ if they belong to the same subspace.

Using the affinity matrix Z, different algorithms have different approaches to segment the data points into clusters. One of the simplest approaches is to threshold and group the points, such as [55]. However, thresholding is sensitive to noise. Boult et al. [39] use a more stable approach by finding eigenvectors of Z and subsequently applying spectral clustering to segment the data. Other algorithms [107, 122, 233] follow the same steps but use an agglomerative approach to tackle the effects of noise.

The advantage of these algorithms is the low computational complexity and low running time. However, the correctness of these algorithms only holds under the assumption of independent subspaces. Specifically, Equation 2.9 and the affinity matrix Z have been proven to be valid only when the subspaces are independent, which is a stricter assumption than disjoint subspaces. Next, we present another group of methods that are able to tackle this problem and find clusters in overlapping subspaces.

Polynomial algebraic methods

GPCA [223] is the most popular algorithm of the polynomial algebraic methods. It aims to find clusters in non-independent subspaces by exploiting the geometric intuition of data in high dimensional space. To this end, by considering the original data space as a union of s underlying subspaces $\{S_i\}_{i=1}^{s}$, finding $\{S_i\}$ is translated to the problem of grouping the normal vectors of these underlying subspaces. GPCA consists of two main steps. First, it fits a polynomial of degree s to the input data. The second step then computes derivatives at the points to derive the normal vectors, in order to determine each subspace and its component points. More details are elaborated below.

Each subspace S_i of dimension d_i can be represented by the equation:

$$b_i^\top x = 0 \tag{2.11}$$

where $b_i \in \mathbb{R}^{d_i}$ is the normal vector of S_i. The original data space, which is now considered as a union of s underlying subspaces $\{S_i\}$, consists of points that satisfy this equation:

$$P(x) = \prod_{i=1}^{s} b_i^\top x = (b_1^\top x)...(b_s^\top x) = 0 \tag{2.12}$$

GPCA makes the assumption that the dimensionalities of all subspaces are equal to d to simplify the problem of finding s, d, and the coefficients of Equation 2.12. In fact, it has been shown [65] that with that assumption, there is a unique polynomial that fits the

data. Derivatives of GPCA find these coefficients using different techniques, such as least squares [223], robust statistics [239], or rank minimization [239]. Without this assumption, determining the polynomials for subspaces having different numbers of dimensions remains a challenge, even when the dimensionalities of the subspaces $\{d_i\}$ are known. Huang et al. [105] propose a method using a model selection technique based on a selection criterion called *minimum effective dimension*.

In the second step, the algorithm uses the polynomial to find subspaces and assign points to each subspace. Specifically, the derivatives of the polynomial at a point $x_j \in S_i$ is equal to the normal vector b_i of that subspace S_i, i.e., from Equation 2.12, $\Delta P(x_j) = b_i$ if $x_j \in S_i$. Intuitively, if a point is known to belong to the subspace S_i, the derivative $\Delta P(x_i)$ reveals the normal vector of S_i, which can subsequently be used to construct the basis. Subsequently, knowing the basis, allocating points to the subspace is trivial. The algorithm proceeds as follows: first, a point is chosen and the derivative of the polynomial at this point is computed to derive the basis of the subspace that accommodates the point. All points that belong to this subspace are then found and removed from the original datasets. The remaining data now only contains points that belong to other subspaces. In the next iteration, another point is chosen and the second subspace is constructed following the same procedure. The process continues until all points are allocated.

Following this procedure, the algorithm is not tolerant to noise and is unable to handle outliers, i.e., the algorithm assumes that every point belongs to a cluster. In addition, the assumption that all subspaces have the same dimensionality might be impractical in real-life applications. It has also been shown [222] that the complexity of the algorithm grows exponentially with the number of subspaces and the dimensionality of each subspace. On the other hand, the algorithm is able to find clusters in overlapping subspaces. Moreover, under the assumption of equal subspace dimensionality, the algorithm has low computational complexity and is able to find the number of subspaces s and the subspace dimensionality d by itself.

Statistical methods

Statistical methods make assumptions on the distribution of data and noise in order to apply different statistical techniques to obtain a better estimate of the subspaces and clusters.

Mixtures of Probabilistic PCA (MPPCA) [213] assumes that both the data and noise in each subspace follow a Gaussian distribution, for which it applies Probabilistic PCA (PPCA) on the data to derive the basis of each subspace. PPCA [246, 48] generalizes classical PCA and is able to handle data with noise. MPPCA then proceeds like K-subspace, by alternating between allocating points to subspaces and refining subspaces until convergence.

The difference is that MPPCA applies probabilistic PCA instead of traditional PCA in each iteration to define the subspace. Therefore, the points have probabilistic assignments to subspaces, i.e., the point x_j belongs to subspace S_i with a probability of $p_{ij} \in [0,1]$ instead of a crisp membership $w_{ij} \in \{0,1\}$. Same as K-subspace, the algorithm is reasonably simple and intuitive. In addition, it is less sensitive to noise and can handle outliers. However, it also inherits multiple limitations of K-subspaces. First, MPPCA needs to know the number of subspaces, and the dimensionality of each subspace. Moreover, multiple runs of the algorithm with different initializations are necessary to ensure the algorithm converges to a global optimum.

Agglomerative Lossy Compression (ALC) [66] also assumes that data in each subspace follows a Gaussian distribution and transforms the problem of subspace clustering into decomposing individual distributions from a mixture of Gaussian distributions of multivariate data. The algorithm finds the optimal segmentation of data that minimizes the coding length, subject to a certain distortion. The algorithm observes that the overall coding length can be minimized in a gradient descent fashion. Specifically, each group is initialized to a point, then they are subsequently merged in a bottom-up fashion. At each iteration, two groups are chosen so that merging them results in the greatest reduction in the coding length, until no further reduction is possible. One advantage of ALC is that it does not require any prior knowledge of the number of subspaces or dimensionality of each subspace. It has also been demonstrated that the algorithm can handle noise and outliers. This algorithm, however, can only find clusters in disjoint subspaces.

Spectral clustering based methods

Sparse Subspace Clustering (SSC) [76] and Low Rank Representation (LRR) [139] are two state-of-the art algorithms that use spectral clustering techniques. In general, these algorithms consist of two steps. The first step obtains the affinity matrix by solving an optimization problem. Specifically, each data point x_i is expressed as a linear combination of the remaining data:

$$x_i = \sum_{i \neq j}^{n} z_{ij} x_j \tag{2.13}$$

where z_{ij} corresponds to the involvement of the point x_j to the linear weighted sum for the point x_i. All coefficients z_{ij} form the matrix Z, and it has been shown that Z can be used to construct the affinity matrix. Specifically, SSC enforces the matrix to be sparse by

regularizing the L_1-norm [136], i.e., to minimize the number of non-zero z_{ij}. The optimization problem of SSC is stated as follows:

$$min \sum_{i \neq j}^{n} ||z_{ij}||_1 \quad \text{s.t.} \quad x_i = \sum_{i \neq j}^{n} z_{ij} x_j \tag{2.14}$$

In contrast, LRR aims to minimize the rank of Z by replacing $rank(Z)$ with its nuclear norm [207] and solving the problem:

$$min \sum_{i \neq j}^{n} ||z_{ij}||_* \quad \text{s.t.} \quad x_i = \sum_{i \neq j}^{n} z_{ij} x_j \tag{2.15}$$

The first step of both algorithms results in a matrix Z that satisfies $z_{ij} = 0$ only if points x_i and x_j do not belong to the same subspace, *under the condition that the subspaces are disjoint and independent*. This is a strong assumption that can be unrealistic for data collected by real life applications, as we will further demonstrate in Chapter 5. The second step applies spectral clustering on the affinity matrix to segment the data into clusters.

These algorithms have been demonstrated to be effective and can handle noise and outliers. However, the approach described above makes the algorithms less scalable to large input datasets. Each additional data point adds one extra instance of the optimization (Equations 2.14 and 2.15) to be solved with a non-trivial computational complexity. We also demonstrate empirically in this thesis that their computational complexities grow rapidly with respect to the number of data points.

Other algorithms of this group include Subspace Segmentation via Quadratic Programming (SSQP) [228], Robust Subspace Clustering [203], and Noisy Sparse Subspace Clustering [230].

2.4.3 Co-clustering Algorithms

As subspace clustering is being discussed, co-clustering (or bi-clustering) is a group of clustering algorithms that cluster high dimensional data that we present in this section. The emergence of these algorithms was motivated by the challenges in clustering DNA micro array data, where each row corresponds to a patient (or sample) and each column to a gene. Studies [114, 126] suggested that only a small subset of genes participate in the genetic pathways that correspond to a certain symptom of interest observed on samples. Therefore, co-clustering algorithms also face the challenge of local feature relevance. These algorithms address this by simultaneously clustering rows and columns. Intuitively, *an ideal cluster*

corresponds to a block of rows and columns that are interrelated [46]. We discuss some representative co-clustering algorithms next, some of which are used as our baselines in Section 6.2. More detailed surveys are available at [31, 174].

Liu et al. [141] propose a *Network-assisted Co-clustering for the Identification of Cancer Subtypes* (NCIS). The algorithm simultaneously groups samples and genes into clusters. Each cluster represents a group of genes that have correlated expressions observed in a group of patients. The algorithm is evaluated with synthetic datasets, as well as large-scale datasets of breast cancer patients. The empirical results indicate the algorithm can cluster patients into different clinically distinct subtypes of cancer and achieve high accuracy on synthetic datasets.

Yu et al. [243] propose a model named *Projective Clustering Ensemble* (PCE) to take advantage of both projective clustering [67, 81] and ensemble clustering [22, 140]. The algorithm first initializes m solutions of projective clustering, i.e., each clustering solution has different assignments of points and dimensions to clusters. These solutions form the base solutions that are ensembled in the next step to achieve a consensus clustering solution, i.e., a solution that maximizes the agreement with all base solutions [93]. The evaluation is performed on eight cancer gene expression datasets and compared with six other clustering techniques. The empirical results demonstrate that cluster quality improves by at least 4.5% compared with other methods. The proposed method is also shown to be robust to noise and input parameters.

Chen et al. [50] propose *Subspace Weighting Co-Clustering* (SWCC) to cluster gene expression data. The method employs a weight subspace matrix to keep track of the contribution and relevance of each gene to each sample. The algorithm defines clustering as an optimization problem to minimize the weighted intra-cluster distance and at the same time maximize the involvement of each gene to each group of samples (if the gene is relevant). The evaluations with synthetic data and ten genomic datasets show that SWCC is a highly efficient co-clustering algorithm that produces comparable or better results than other state-of-the-art clustering algorithms. The baselines of SWCC are the co-clustering algorithms EWKM [116], BBAC [23], ITCC [70], FFCFW [214], and HICC [52], which will be discussed next and will also serve as the baselines of our evaluation.

Entropy Weighting k-Means (EWKM) [116] is built on top of k-means to address the problem clustering high dimensional data. The authors extend the k-means algorithm by introducing the weight of each dimension in each cluster, and subsequently include the entropies of the dimension weights of each cluster in the objective function. The optimization problem therefore simultaneously minimizes the intra cluster dispersion and the absolute value of weight entropies. To that end, the weights of dimensions are adjusted in each cluster,

i.e., irrelevant dimensions have less weight. In addition to being evaluated with synthetic data set, EWKM has also been shown to perform well with real-world News group datasets[11], and with gene expression datasets [50].

Bregman Block Average co-clustering algorithm (BBAC) [23] formulates co-clustering as a problem of matrix approximation. Each result of co-clustering corresponds to an approximation matrix, and the clustering quality is determined by the approximation error. The algorithm proposes that the approximation error can be measured by different loss functions from the class of Bregman divergences (or Bregman distances) [74, 198]. Some popular distances that belong to this class include the Euclidean distance and the KL-divergence distance. A novel contribution of BBAC is the proposal of the Minimum Bregman Information (MBI), which is a generalization of *maximum entropy* and *standard least squares* principles for matrix approximation. BBAC, with the flexibility of using a large class of loss functions, can be considered more than just a single algorithm. In fact, it generalizes ITCC [70], which has the same process and specifically uses mutual information to construct the loss function. Banerjee et al. [23] also point out that by choosing different loss functions the proposed algorithm is equivalent to the well-known k-means or MSSRCC [53] algorithms.

Flexible fuzzy co-clustering with feature-cluster weighting (FFCFW) [214] is a fuzzy co-clustering algorithm that resolves one of the constraints shared by previous algorithms on the equal number of clusters and number of subspaces. FFCFW solves an optimization problem that simultaneously minimizes the error in grouping dimensions into cluster subspaces, minimizes the grouping of points, and adjusting the weights that control the association between each point and each dimension. FFCFW has been demonstrated to work with different real-world text and document datasets such as News groups[1], and WPD1 [199], as well as gene expression datasets [50].

HICC [52] is a hierarchical co-clustering algorithm that constructs hierarchical structures of both the points (rows) and dimensions (columns). The algorithm takes advantage of the monotonicity of the mutual information of co-clusters, and uses a greedy divisive strategy to maximize the mutual information between the row clusters and column clusters, which is used to monitor the quality of the clustering result. The algorithm starts with one row cluster and one column cluster that contain all the rows and all the dimensions of the original dataset respectively. The greedy strategy then iteratively performs splits of clusters to maximize the mutual information between row and column clusters. Note that in each iteration either row clusters or column clusters can be split. The algorithm terminates when the desired numbers of row clusters and column clusters are achieved, or when the mutual information reaches a provided threshold. The paper also presents extensive evaluation

[1] https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups

and demonstrates its superior performance compared to other co-clustering algorithms. In addition, the hierarchical structure of the rows and columns also allow the clustering results to provide insights in the relationships between different clusters.

2.5 Large Volume and High Dimensional Data

With the definition and conception of *high dimensionalities* and *large volumes* constantly evolving, we aim to discuss the current state and clarify these terms that we use in this thesis. Our discussion is from the perspective of subspace clustering applications. We first discuss the volumes and dimensionalities of the data used in different clustering algorithms in this section, before stating the scale of inputs we aim to address in this thesis.

We summarize different aspects of the representative algorithms in Table 2.1. The largest volumes and highest dimensionalities of the inputs, either synthetic or generated by real world applications, upon which these algorithms were demonstrated are also summarized.

Most of the traditional algorithms that use bottom up and top down clustering approaches can handle large inputs that have up to 500,000 data points. However the datasets used in these experiments in general have no more than 100 dimensions. Although some of these experiments are dated and it can be argued that they can scale to inputs that have much higher dimensionalities with more contemporary computing resources, the fundamental challenge is that the time complexities of many of these algorithms grow quickly w.r.t the number of dimensions. For example, the time complexity of CLIQUE is exponential in the number of dimensions; SUBCLU has been shown to have at least quadratic complexity with respect to the number of records, dimensionalities of inputs and dimensionalities of clusters; the time complexity of ENCLUS in the worst case is $O(n \times d + m^d)$ with m being the number of bins into which each dimension is divided [51]. Accordingly, recent studies that attempt to improve the performance of these algorithms do not scale to significantly higher dimensionalities. More specifically, CLUS that uses Spark to parallelize SUBCLU requires more than 30 minutes to cluster data that has 20 dimensions in its experiment. Goyal et al. [89] propose a parallel framework to speed up CLIQUE, ENCLUS and MAFIA. The framework is evaluated with huge volume datasets, but the number of dimensions does not exceed 14. MR-MAFIA is the only algorithm of this type that is shown to work with high dimensional data. The authors use Map Reduce with a 16-core master and a cluster of 30 nodes to cluster datasets that have 15,000 data points and 5,000 dimensions in about three hours [86].

The group of more recent subspace clustering algorithms that are based on spectral clustering and matrix factorization techniques, such as SSC, LRR and ALC, can handle much

Table 2.1 Summary of representative subspace clustering algorithms.

	Non-disjoint subspaces	Scalable to large inputs (max input volume)	Max input dimensionality (associated input volume)	Clusters with varying density	Clusters with varying dimensionality	Other assumptions/requirements/notes
CLIQUE [14]	Y	Y (500K)	50 (100K)	N	Y	Same grid size/density threshold in all dimensions.
ENCLUS [51]	Y	Y (500K)	30 (volume not specified)	N	Y	Same grid size/density threshold in all dimensions.
CLTree [138]	Y	Y (500K)	20 (500K)	Y	Y	High computational complexity
SUBCLU [120]	Y	N (275K)	50 (volume not specified)	N	Y	High computational complexity
PROCLUS [13]	Y	Y (500K)	20 (500K)	N	N	
FINDIT [232]	Y	Y (5000K)	50 (100K)	N	N	
PROFIT [182]	Y	Y (2310)	1958 (925)	N	N	
K-Subspaces [11]	Y	Y (50K)	50 (50K)	N	Y	Number of clusters and dimensionality of each subspace must be given
GPCA [223]	N	N (800)	3 (192)	Y	–	Number of clusters and dimensionality of each subspace must be given
MPPCA [213]	Y	Y (2025)	64 (500)	N	–	Number of clusters and dimensionality of each subspace must be given Data and noise follow Gaussian distribution
ALC [66]	N	N (1000)	64 (1000)	Y	–	Number of clusters and dimensionality of each subspace must be given Data follows degenerate Gaussian distribution
SSC [76]	N	N (2432)	2016 (2432)	Y	Y	
LRR [139]	N	N (640)	2016 (640)	Y	Y	
SSQP [228]	N	N (320)	1024 (320)	Y	Y	
SSWC [50]	N	Y (1000)	12800 (1000)	Y	Y	

higher dimensional data, e.g., SSC has been demonstrated to work with data that has 2016 dimensions in its application to clustering face images of size 48×42 pixels. However, the inputs of these applications have small volumes as shown in Table 2.1. A common technique shared by these algorithms is to perform point grouping by solving an optimization problem for each data point, which is computationally expensive. Hence, it is sensible that they do not scale well to inputs having large volumes, as we elaborate and demonstrate further in Chapter 5 of this thesis.

SWCC is able to cluster large inputs of high dimensional data. Although the scalability to large volume inputs are not discussed in [50], we demonstrate later in Section 5 that SWCC can scale to very large inputs.

Next, we further discuss the problem of clustering high dimensional data from the perspective of applications and with a focus on the size of the data these applications usually need to handle. Based on this literature, we identify the expected limits on the sizes of inputs that our technique should be able to handle.

In applications of the Internet of Things, the data collected by a sensor device usually forms a single dimension in the dataset. The dimensionalities of data collected by these applications therefore depends on the deployment scale and number of devices, which can vary from a few dozens [211, 1, 150, 177] to thousands [7, 8]. Among the *experiments of clustering on real-world IoT data* that we can find, the largest volume we record is 28,550 (10-dimensional data) [211] and the highest number of dimensions is 449 [177].

In applications of image processing, each image can be "flattened" and represented as a vector of pixels. The dimensionalities therefore depend on the resolution of the images. To this end, depending on the quality of the images being used, the dimensionalities can vary in the range of a few thousands (e.g., MNIST [63] and CIFAR [44]) to hundreds of thousands (e.g., ImageNet [62] and CalTech-256 [90]). These are examples of only moderate quality images. For higher quality images, the number of dimensions of the vector of pixels can grow to the millions. However, we are unable to find any clustering algorithm that was demonstrated to work with such large images. In fact, a popular practice in clustering images is to first reduce the number of dimensions. For example, Mathilde et al. reduce the image inputs to 256×256 [42], and Ehsan et al. reduce the 192×168 images to 48×42 images for their clustering experiment [76].

In bioinformatics, each gene usually corresponds to a dimension, and each patient or sample from which data is collected corresponds to a data point. The datasets generated by bioinformatics applications can have low volumes (low numbers of samples), but the numbers of dimensions can grow to several thousands [18, 71, 161].

There are different perspectives on high dimensionalities in applications of natural language processing. A popular approach to process the textual data is to encode the words using a pre-trained word embedding that can capture the semantic relationships between words. Popular pre-trained embeddings include GloVe [169], fastText Wiki [153] and ELMO [170], whose dimensionalities are usually set to less than 1000. In contrast, applications that analyze the data on the document level use the presence or absence of a word in that document as a feature. To this end, the dimensionality of a data point depends on the size of the corpus vocabulary, which is determined by the frequency threshold used to retain only the most popular words. The corpus vocabulary varies in different applications and can be up to 100,000 as summarized by Chen et al. [49].

Our initial motivation and focus area in this thesis is to use subspace clustering to analyze data generated by applications of IoT. Nevertheless, as the field of IoT evolves, the sizes and dimensionalities of such datasets keep growing rapidly [147]. To this end, in addressing Research Question 3, we consider the existing applications of clustering in the literature as discussed above to set the expected scalability of our technique. To this end, *we aim to be able to cluster datasets that have up to thousands of dimensions and hundreds of thousands of data points.*

2.6 Summary

In this chapter, we discussed the motivations and challenges of subspace clustering. We also presented multiple algorithms with respect to their approach, their scalability to large inputs, and their ability to handle noise and outliers in the data. Table 2.1 provides a summary of the main methods that we have surveyed in this chapter. Although using different techniques and strategies, the limitations of these algorithms can be summarized as follows:

- Finding clusters in overlapping subspaces remains a challenge. Although recent algorithms can find subspace clusters effectively, they make the assumptions that the subspaces are disjoint, which can be an impractical assumption for data collected from real-life applications. The algorithms that are able to tackle this problem make other impractical assumptions about subspace dimensionality, e.g., that the number of dimensions of all subspaces are equal.

- Model selection is a major problem and it can be challenging to provide the required parameters to the algorithms in some cases. Examples include the dimensionality of each subspace, the grid size or the density in each dimension. This second limi-

tation, together with the first one, can make the algorithms less practical in real-life applications.

- Many algorithms that use sophisticated optimization techniques are unable to handle large inputs. For example, matrix-factorization-based and spectral clustering-based algorithms do not scale well to large input volumes; algorithms using bottom-up and top-down approaches also take a considerable amount of time to cluster a relatively small dataset [160].

We aim to address these research challenges to propose a novel clustering algorithm that can find clusters in non-disjoint subspaces and can scale to large inputs while having relaxed requirements on the model selection. We also aim to avoid impractical assumptions on the subspaces or the structure of the data.

We begin in the next chapter by presenting our first contribution in using clustering to solve a real-life application of the Internet of Things. This motivating work demonstrates the challenges in clustering multi-dimensional data as described in the literature, and emphasizes the need for subspace clustering algorithms to address these challenges.

Chapter 3

Profiling Pedestrian Distributions and Anomaly Detection in a Dynamic Environment

One field that has observed a proliferation of high dimensional data is the Internet of Things (IoT), in which the development of data capturing technology has resulted in increasing numbers of sensors being deployed and producing large data sets that are high dimensional. We start our research by analyzing data collected by a real life IoT application in urban monitoring in order to study the practical attributes of high dimensional data as well as the accompanying challenges. Specifically, we model the pedestrian distributions of the City of Melbourne, Australia, and use the model to detect anomalous events. We first start the study with a simple problem of clustering and anomaly detection on a small dataset of two dimensions to gain a better understanding of the data and the problem. After achieving good results in this experiment, we demonstrate the infeasibility of scaling the same approach to higher dimensional data. Specifically, as we apply the same approach to 3-dimensional data with different parameters, the technique can only find either a very small or very large cluster that does not offer any useful insights on the pedestrian distributions.

We then present a separate technique that is capable of detecting anomalies in datasets of up to 10 dimensions during certain periods of the day. We illustrate the attributes of local feature relevance in data of a practical application, and then draw an analogy between our proposed approach and the problem of subspace clustering.

While this chapter presents our first contribution in analyzing data collected by a real-life application of the IoT, the main contribution to the thesis is to motivate the needs for (1) a method that can address local feature relevance in high dimensional data to measure the

similarity between data points, and (2) the necessity of using subspace clustering to analyze high dimensional data more effectively.

3.1 Introduction

The emergence of the Internet of Things (IoT), which is a major evolution of the current Internet into a network of interconnected objects, has resulted in many sensors being deployed to monitor various measures of interest in a city [3, 92, 115, 196]. Examples of such deployments include SmartSantander [8], the City of San Francisco, USA [7] and the City of Melbourne, Australia [1]. Pedestrian activities play an important role in the dynamics of cities. The involvement of pedestrians in almost every operation of the city, such as infrastructure, transportation and traffic, has made understanding their distribution an essential task for architects, city planners or event schedulers. The pedestrian movement data collected using sensors in the city enable the pattern of movements of pedestrians to be studied in a systematic way. However, due to the volume and complex structure of the data, there has been limited progress in utilizing sensor data to model pedestrian behaviour and detect any interesting events.

In this study, we aim to characterize the normal behaviour of pedestrian flows in terms of correlations in the load profile by time of day across different locations. This knowledge can be used by city planners in catering for pedestrian flows, in assisting businesses to target their services according to demand, or in understanding the dynamics of cities. We then utilise the model to detect anomalous pedestrian activities, which helps to characterize and forecast the impact of special events on pedestrian activities.

3.1.1 Data

The data used and discussed in this chapter is the **Pedestrian Counting System of the City of Melbourne**. Since 2012, the City of Melbourne has installed an automated system to count the number of pedestrians at strategic locations in the central business district (CBD) of the city. Each sensor is installed on a street pole or under an awning to form a counting zone on the footpath below [1]. The sensors count the number of pedestrians from multiple

The main content of this chapter has been published in the following papers:

M. T. Doan, S. Rajasegarar, M. Salehi, M. Moshtaghi, and C. Leckie. Profiling pedestrian distribution and anomaly detection in a dynamic environment. Proceedings of the 24^{th} ACM International on Conference on Information and Knowledge Management (CIKM), 2015, pp. 1827-1830.

M. T. Doan, S. Rajasegarar, and C. Leckie. Profiling pedestrian activity patterns in a dynamic urban environment. Proceedings of the 4^{th} International Workshop on Urban Computing (UrbComp), 2015.

directions, and regularly update the data to a central server, where they can be visualized and downloaded. The devices and the mechanism of the counting sensors are shown in Figure 3.1. The full list of sensor deployment locations is described in Table A.2 in the Appendix.

We also propose various ground truth labels based on our understanding of pedestrian activities in Melbourne to quantitatively evaluate and compare our methods. The first set of ground truth labels describes the normal types of distributions of pedestrians at different times of the day (Table 3.5), which is used to evaluate our clustering algorithm. The other ground truth labels contain the city events during which pedestrian activities are considered anomalous. These sets of ground truths are summarized in Tables 3.9, 3.16, and 3.17.

3.1.2 Methodology

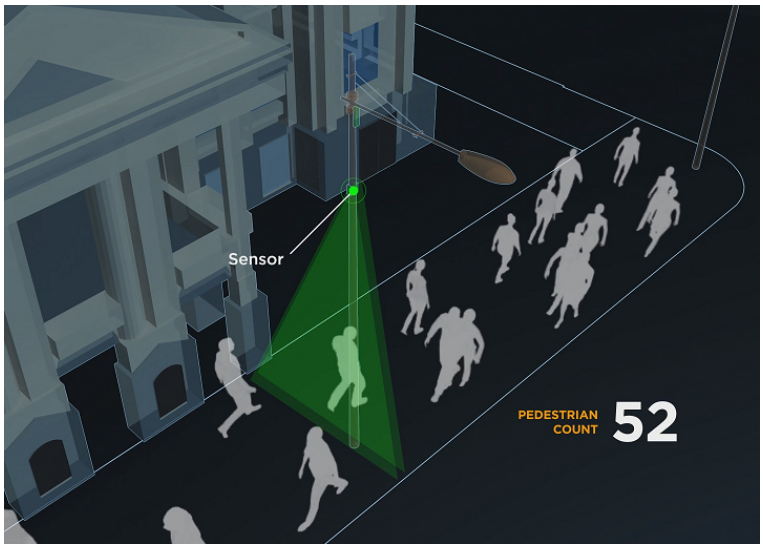
We briefly present two separate approaches to model the pedestrian distribution as follows. The details of each method are provided in the subsequent sections.

Clustering-based approach

First, we use clustering to model the pedestrian distribution and use it to detect anomalies. By analyzing one year's worth of data we aim to construct a reliable and meaningful model of pedestrian movements. This raises two key research challenges: (1) the data is periodic on daily, weekly and monthly cycles, hence looking only at the sequence of data might hide important patterns, and (2) the quality of clusters can decrease when input data is collected over longer time scales, since the models of normal behaviours may vary throughout the year. Therefore, it is essential to identify different normal states of the system, as well as modelling the switching of the system between these states, in order to effectively model the pedestrian distribution and accurately detect anomalous conditions.

We examine two techniques in this study. First, we use HyCARCE [156] to model the pedestrian distribution. HyCARCE is a computationally and memory efficient clustering technique that uses hyperellipsoids for clustering. It performs clustering on the entire input dataset in a single pass. We then use the clustering result to detect anomalous activities. Here, we assume any data points that do not belong to a cluster are an anomaly. The second technique used is the Ensemble Switching Model [190], which is an anomaly detection technique that can cope with dynamic environments where the system switches between different states intermittently. These two techniques are validated against known events, and evaluated based on their capabilities of correctly identifying anomalous activities.

Our key contribution is that we demonstrate that it is feasible to model complex urban pedestrian counting data. Furthermore, we evaluate whether the accuracy of the Ensemble



(a)

(b)

Fig. 3.1 (a) Operation of a counting sensor. Figure reproduced from [1]. (b) A pedestrian counting sensor. Figure reproduced from [6].

Switching Model outweighs its complexity compared to the static clustering model of HyCARCE for modelling pedestrian movements and detecting anomalies.

Frequent pattern-based approach

While the previous approach gives good results in clustering and anomaly detection, we are not able to scale it to higher dimensional data to include data from more locations. Therefore, we aim to extend the previous approach and model the activities of pedestrians over multiple locations in the CBD of the City of Melbourne simultaneously using a different technique.

Fig. 3.2 Deployment map of pedestrian counting sensors, where major locations are labelled. Figure reproduced from the website of the Pedestrian Counting System of the City of Melbourne [1].

Table 3.1 An illustrative example of transactions and items for four locations during n timestamps.

Transactions	
T_1	$profile^{(1)}_{flinders}$, $profile^{(1)}_{southerncross}$, $profile^{(1)}_{princebridge}$, $profile^{(1)}_{artcenter}$
T_2	$profile^{(2)}_{flinders}$, $profile^{(2)}_{southerncross}$, $profile^{(2)}_{princebridge}$, $profile^{(2)}_{artcenter}$
...	...
T_n	$profile^{(n)}_{flinders}$, $profile^{(n)}_{southerncross}$, $profile^{(n)}_{princebridge}$

By mining the correlations of the pedestrian distributions across multiple key locations of the city, we define a profile of the normal pedestrian distribution, and then use this model to detect any anomalous pedestrian distributions.

We employ a frequent pattern mining [37] based approach to reveal the correlation of pedestrian counts at various locations in the city during a given time of the day. To this end, each hourly *profile* of pedestrians at a location is considered as an item; the profiles of pedestrians at all locations in an hour (i.e., a snapshot of pedestrian distribution across the CBD at a certain timestamp) form a transaction. Table 3.1 illustrates the concept of transactions and items in our context.

Frequent pattern mining by default only works with categorical data [37], hence each item in the transaction table is not the raw hourly count of the pedestrians. Instead, the

raw counts are discretized into categories, which are referred to as *pedestrian profiles* and represented as $profile$ in Table 3.1. We use three categories as described in greater detail in Section 3.4.3. Note that this method allows us to work with missing data without the need of backfilling. For example, if the sensor at the Art Center is faulty and does not provide the count at T_n, it does not appear in the transaction. If this does not happen frequently, it does not affect the frequent patterns being mined.

To this end, each frequent pattern being mined represents a distribution of pedestrians that are repeatedly observed at locations indicated by the items of that frequent pattern. By constructing the set of frequent patterns that represent the majority of the observations in the dataset, we build a model that describes the normal state of the pedestrian count distribution. Subsequently, we use this model to detect any anomalous distributions of pedestrians. There are two main research problems that need to be addressed in this approach. First, the data is multidimensional and the raw values of pedestrian counts in each dimension (location) might carry a different meaning in terms of the distribution when compared with the other dimensions, e.g., the count of pedestrians at a busy train station might have a different significance than the same value at a vacant park. For example, an hourly count of 2,000 pedestrians at a train station can be considered normal while the same value being observed at a vacant park might indicate an anomaly, e.g., an event. Second, modelling the pedestrian distribution is not purely a task of frequent pattern mining. In addition to producing reliable frequent patterns with high support, the algorithm needs to ensure that the set of frequent patterns are representative of not only the points in the dataset but also of the locations being investigated. For example, if the algorithm just extracts very short and simple patterns (with a small number of items), it is very likely that the patterns are shared by most of the transactions. Such patterns in this case apply to most of the times in the input (transactions), but only cover very few locations (items), and hence cannot provide a holistic presentation of the data. For example, out of 44 locations in the CBD, if the algorithm just extracts the patterns that consist of two locations, the model does not reflect the distribution at other locations and cannot be considered a good representation of the data. On the other hand, producing highly complicated patterns might divide the input into multiple groups with excessive granularity and consequently fail to summarize the intuition of the results. Therefore, only by fulfilling these two requirements will the model be likely to be meaningful and the anomaly detection accurate.

3.2 Related Work

Multiple studies have analyzed user-centric pedestrian activities by tracking the path of pedestrians using GPS or ubiquitous devices in historic districts, shopping malls or their work commute, such as [26, 154, 221]. However, there are few studies on the crowd behaviour of the city as a whole.

The Downtown Raleigh Alliance (DRA) conducted a study on pedestrian activities [5] in key downtown corridors in order to better understand pedestrian preferences, evaluate and plan pedestrian infrastructure and promote local commerce, tourism and liveability. The study relies on manually collected data using a nationally-vetted pedestrian count methodology. The data is collected periodically, with a focus on the morning peak, lunch time, afternoon peak and late night in three target areas. Simple statistics and data visualisations are used to analyze the overall trend, the total volume and the average number of pedestrians per hour. Although the study extracts useful information for the city government, it does not construct pedestrian load profiles throughout the day. Therefore it is not trivial to extend the knowledge to other applications in a systematic way. Moreover, the study is conducted over a relatively short period of four weeks, which limits the ability to model and analyze the effects of external factors such as weather, seasons, and public holidays throughout the year.

There have also been multiple studies using location based social networks, e.g., Facebook, Twitter, and Flickr to understand crowd distributions [82, 103, 145]. For example, Ferrari et al. [82] utilize the Twitter data of Manhattan-based users to model the crowd footprint of particular locations and extract urban patterns. By associating the crowd footprint to the words used in messages, Ferrari et al. use Latent Dirichlet Allocation [33] to reveal trends and routing behaviour in the city for each day of the week. However, a limitation of this study is that the number of topics, corresponding to the different pedestrian load profiles, needs to be predicted and provided in advance. Moreover, the model assumes the pedestrian load profile will not vary throughout the one-year span of the collected data. This assumption is not realistic and may affect accuracy. For example, by validating all Mondays with a typical Monday profile, the accuracy yields less than 40%. One sensible explanation is that pedestrian activities are likely to vary at different times of the year, e.g., pedestrians are more active during the night time in summer than in winter, or much more active during festival seasons.

Kaltenbrunner et al. [121] study the usage of public bike sharing services across multiple locations in the City of Barcelona. By investigating the number of bikes rented and available at the bike sharing stations, the study infers the cycling activities, analyzes the mobility patterns and models the cycling activities of Barcelona's population. The study uses seven

weeks of data, which are directly retrieved from the service provider. When analyzing the data at each location separately, Kaltenbrunner et al. distinguish two different profiles for weekdays and weekends and justify these patterns with routines in the city, such as working hours, lunch breaks, etc. The study also analyzes the data on a macroscopic level and reveals the global mobility pattern of Barcelona's cyclists. The result is expressed as a geographic heat map which presents the increase and decrease of cycling activities over different times and locations in the entire city. This result, however, can only be interpreted visually and does not really facilitate further analysis in a systematic and programmatic way. In addition, the metrics used in this study to deduce the profiles are fairly simple, e.g., average, standard deviation, local maximum and local maximum. We believe more complex analysis on this type of data can reveal more detailed patterns.

All the studies mentioned above either use a short data collection period or assume a constant state of the pedestrian load profile, which we believe hides useful information and results in false detection of anomalies. In this chapter we use different techniques to model the pedestrian distribution data in a dynamic environment over a long period.

3.3 Clustering-Based Approach

3.3.1 Problem Statement

Let X denote a set of N observations from time t_1 to t_N: $X = \{x_i : i = 1..N\}$ where $x_i \in \mathbb{R}^d$ is a vector of d dimensions corresponding to the pedestrian count values observed at d different locations at time t_i.

First, we aim to find a set of c clusters $\mathscr{C} = \{C_k : k = 1..c\}$. Each cluster C_k contains data points $\{x_i\} \subset X$ that represent a certain pattern of the pedestrian distribution that is observed at times $\{t_i\}$. From the cluster set $\mathscr{C}$, we extract a set A_C that contains all anomalies that do not belong to any clusters, i.e., an anomalous pattern of pedestrian activities that are not commonly observed. Here, $A_C = X - \bigcup_{k=1}^{c} C_k$. This approach provides a static model of the underlying system, which does not capture how the clusters change over time. Next, we develop a clustering model that can cope with nonstationarity, or systems that switch between states over time. To this end, we use an Ensemble Switching Model [190] and aim to find the set of anomalies A_{ES} from a set of observations X that switches between different states. By evenly dividing X into windows of length w, we identify the different states that exist in the data at different times. As time progresses and the system switches between states, we select only the relevant states as the basis for anomaly detection.

By validating the performance of each approach against known events and by comparing A_C and A_{ES}, we seek to evaluate to what extent the dynamic Ensemble Switching Model provides greater accuracy in detecting anomalies in comparison to static clustering, given the extra logic and resources required to model switching data streams.

3.3.2 Methodology

Static Clustering HyCARCE

Hyperellipsoidal Clustering for Resource Constrained Environment (HyCARCE) is a grid-based clustering algorithm with low computational complexity of $O(N)$ [156, 180]. The algorithm can be summarized in four steps as follows:

Input: A dataset of N d-dimensional records $X = \{x_i : i = 1..N\}, x_i \in \mathbb{R}^d$

Ouput: A set of c clusters $\mathscr{C} = \{C_k^{X_j} : k = 1..c\}$ (c is determined algorithmically)

Step 1: Divide the input space into a set of fixed-size d-dimensional cells and prune the cells with low density.

Step 2: Fit a hyperellipsoid into each cell that covers at least 95% of the cell data. The hyperellipsoid is defined by:

$$(x-\mu)^T \Sigma^{-1} (x-\mu) = (\chi_d^2)_{0.95}^{-1}$$

where μ and Σ are respectively the mean and covariance of the data in the cell. Σ^{-1} defines the characteristic matrix of the hyperellipsoid.

Step 3: Enlarge each hyperellipsoid by scaling the characteristic matrix to accommodate new data points. After each enlargement the mean μ and covariance matrix Σ are recomputed. Enlargement and re-adjustment are repeated until the change in the number of data points being added to the new hyperellipsoid is smaller than a predefined threshold.

Step 4: Remove redundant hyperellipsoids. If the distances between the centres of two hyperellipsoids are smaller than a threshold, the one with a lower number of data points is considered to be redundant and removed.

Studies [156, 180] have shown that HyCARCE produces equivalent or better accuracy compared to popular clustering algorithms such as DENCLUE[102], k-means [236] and GK [94].

Ensemble Switching Model for Anomaly Detection

The *Ensemble Switching Model* (ESM) [190] is a dynamic anomaly detection technique that addresses the problem of the underlying system switching between multiple states in the

stream of data. By evenly dividing the input data stream into small windows, it constructs a model of the states corresponding to normal behaviours, and uses this dynamic model to detect anomalies, rather than using the static model of HyCARCE.

Step 1 (windowing): The data stream is divided into windows of size w. The w observations of window k are then clustered using HyCARCE into a set of hyperellipsoids that form the clustering model C_k of that window.

Step 2 (weighting): The distances between the clustering model of the current window C_k to previous windows $\{C_1, ..., C_{k-1}\}$ are computed. The distance between two models is defined as the sum of the focal distances [155] between all pairs of corresponding clusters that match between the two models. The relevance between clustering models are then deduced from the distances, which is then used in the final step to score anomalies.

Step 3 (anomaly scoring): The detection of whether an observation is an anomaly is calculated using the clustering models of both the current and previous windows. The relevance weighting computed in **Step 2** ensures only relevant clustering models take part in the scoring.

3.3.3 Experiments

In this section, we present the experimental settings to perform HyCARCE clustering, as well as the results of anomaly detection on the pedestrian counting data using both HyCARCE and Ensemble Switching Model.

In this initial work on data collected by a real-life application, we focus on understanding the data as well as evaluating anomaly detection, without prioritizing the scalability of the experiment to higher dimensional or larger volumes of data. To that end, we first focus on two adjacent locations in the CBD of Melbourne for analysis: Flinders St-Elizabeth St (East) (location 7 - we will refer to this location as Flinders Street in this experiment for simplicity) and Princes Bridge. These two monitoring points are located at the heart of the CBD and are representative of the movement of pedestrians between the CBD and South Melbourne throughout the day. The time period of the monitored data is from 1^{st} January to 31^{st} December 2014. There is a pedestrian count each hour at each location, which means there are a total of 8760 pedestrian counts for each location. In this experiment we exclude the pedestrian counts that were collected during weekends, resulting in 6264 records for each location. We understand that the pedestrian activities during weekends in the CBD of Melbourne significantly differ from those during weekdays. Due to the lack of business and office activities, and the reduced frequency of public transport on weekends, weekend pedestrian counts are usually low during hours when they are observed to be high during weekdays. On the other hand, the weekend activities (such as street performances, sport

Table 3.2 Details of the input data.

Number of instances	6264
Number of dimensions	2
First date	1^{st} Jan 2014
Last date	31^{st} Dec 2014

events) also lead to more pedestrian activities during 10am-11am and 2pm-5pm, when lower pedestrian counts are usually observed during weekdays. The differences in pedestrian activities are also observed on Saturday nights. For these reasons, we found that including weekends in the input reduces the homogeneity of the hours in a cluster, and subsequently makes the analysis of results harder. This, together with the small set of low dimensional data, can be considered as a limitation of this study. However, it still allows us to gain insights into the data, the challenges of analyzing multi-dimensional data, and subsequently allows us to address and fix these limitations, as presented in Chapter 6.

These counts form the dataset X to be clustered, where each data point $x \in X$ is a two dimensional point representing the pedestrian counts at the two locations at a certain time. The details of X are summarized in Table 3.2. Our aim is to detect the abnormal data points, i.e., to identify time points when the pedestrian counts at the two locations are abnormally different from those generally observed in the dataset.

Since the count data is not evenly distributed, with a large portion of the observations being skewed towards the low values, the data is preprocessed to reveal its structure more clearly, the steps of which are illustrated in Figure 3.3. First, instead of using the absolute counts of pedestrians, we used the differences between successive counts. These values represent the hourly changes in the number of pedestrians. Second, we apply a polar transformation to separate the values into different segments and to achieve a more even distribution. Finally, the data was normalised to the range $[0, 1]$. The pre-processed data is then given to HyCARCE and ESM.

3.3.4 Evaluation of HyCARCE Clustering

The experiments are evaluated in different stages. First, as HyCARCE is a clustering algorithm, we evaluate its ability to produce meaningful clusters. Here, a meaningful cluster contains records that feature similar characteristics of pedestrian activities. For example, we consider a cluster that mostly contains records of 8am, 9am, 5pm, 6pm as a meaningful cluster as all these hours correspond to the morning peak and afternoon peak, where the numbers of pedestrians are expected to significantly increase compared to other times of the day. The effectiveness of HyCARCE is evaluated both qualitatively and quantitatively:

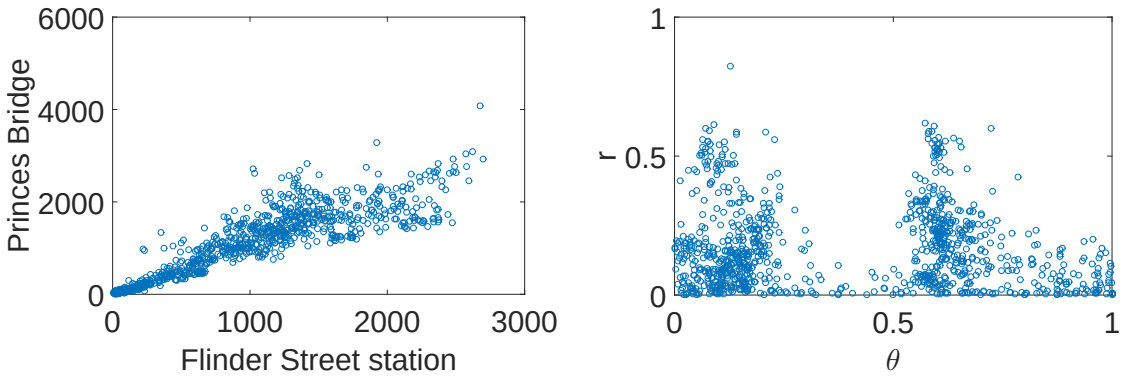


Fig. 3.3 Data preprocessing steps: (a) raw count values, (b) polar transformation and normalization to range $[0,1]$ ((θ,r) are the polar coordinates)

- **qualitatively**: Each cluster is inspected and its components are matched to the types of activity of the day, e.g., morning peak, lunch time, etc. We then analyze and discuss if each cluster features a distinguished characteristic of the day.
- **quantitatively**: We use our best understanding of the pedestrians of Melbourne to propose in advance the ground truth that labels the types of pedestrian activities at different times of the day. The ground truth is summarized in Table 3.5 and elaborated in the relevant section. We then compute the precision, recall, and F1 score for the clustering results.

Since HyCARCE is also the underlying clustering method of Ensemble Switching, it is necessary to verify its effectiveness. To this end, we show the HyCARCE clustering results for both one week's data and one year's data.

HyCARCE requires only one parameter [156], which is the grid size g to segment points in the data space into cells. As our input has been pre-processed and normalized to $[0,1]$, we performed HyCARCE with the 15 values of $g \in \{0.01, 0.02, 0.03, ..., 0.14, 0.015\}$. The clusters are then evaluated based on their sizes and their hourly components. We achieve the most meaningful clusters with $g = 0.08$ and present the corresponding qualitative and quantitative evaluation next.

Qualitative evaluation

HyCARCE successfully models the distribution of pedestrians when applied on one week's worth of data. The clusters are presented in Table 3.3; each tuple represents the hour of the day that corresponds to the data point, as well as its frequency. For example, HyCARCE

groups 7*am* of all five days of the week, i.e., $(7,5)$ into cluster C4W. Due to the noisy nature of urban data [25], it is not realistic to expect that the clusters will not contain multiple records of other random hours as noise. In this experiment, to keep the table concise and not having multiple small groups, we only show components whose frequency is larger than 30% of the input volume. For example, when clustering one year's data, we only display components that appear more than 75 times (30% $\times$ 252 weekdays) in a cluster. Note that this is only for presentation, and our quantitative evaluation uses the entire clustering results.

Figure 3.4a shows that the data points are clustered into visually clear and well separated clusters. Looking into the records of each cluster, we observe that each cluster corresponds to the distributions of pedestrians at different times of the day. Specifically:

Table 3.3 HyCARCE clusters of one week's worth of pedestrian counting data

Clusters	(Hour of day, Instance count)
	One week's data
C1W	(22,5), (23,5), (0,5), (2,2), (19,2)
C2W	(2,2)
C3W	(5,5)
C4W	(7,5)
C5W	(6,5), (17,4), (11,3)
C6W	(8,4), (12,4), (16,5)
C7W	(9,5), (18,5)
C8W	(1,4)
C9W	(13,4), (20,2), (21,3)

Table 3.4 HyCARCE clusters of one year's worth of pedestrian counting data

Clusters	(Hour of day, Instance count)
	One year's data
C1Y	(18,225), (9,239), (19,108)
C2Y	(7,233), (8,196)
C3Y	(16,180), (6,229)
C4Y	(5,253)
C5Y	271 records of random hours of different days
C6Y	(14,187), (20,220), (21,135), (22,191), (23,243), (0,226), (1,258), (2,191), (19,139)
C7Y	5 records of random hours of different days

- Cluster **C7W** characterizes the distribution during the morning peak (9am) and afternoon peak (6pm).

- Cluster **C6W** characterizes the distribution during the morning peak (8am), lunch time (12pm) and 4pm. Given that cluster **C7W** also describes morning peak, this result suggests that it offers a more granular comparison that the distribution of pedestrians at 8am is different from that at 9am (to our understanding, this area of the CBD is more crowded at 9am - which is the highest peak hour in the morning.)

- Cluster **C9W** characterizes the distribution in which pedestrian activities start to decrease from the peak hours: 1pm (after lunch time), 8pm-9pm (after evening peak/dinner time). We also observe the lack of the morning hours (10am) in this cluster. Analyzing the results shows that the pedestrian counts at 10am are scattered in five different clusters. The lack of presence of these morning hours are also observed when we cluster one year's worth of data, which is also shown in Table 3.3. Without further investigation, our assumption is that the pedestrian activities at 10am are more variable from day to day, and do not form a sufficiently strong pattern to form a cluster.

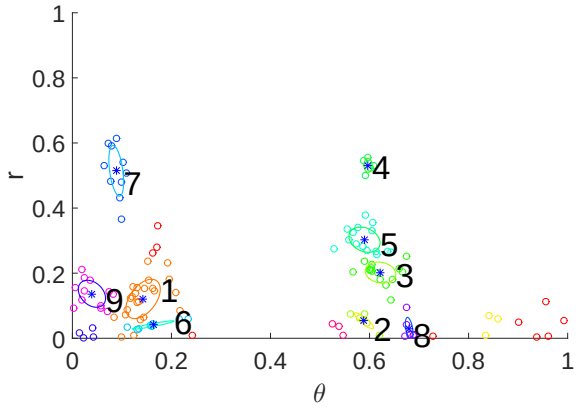
- Cluster **C2W** and Cluster **C8W** correspond to very early hours after midnight (1am and 2am)

- Cluster **C3W** and Cluster **C4W** correspond to early hours in the morning (5am and 7am)

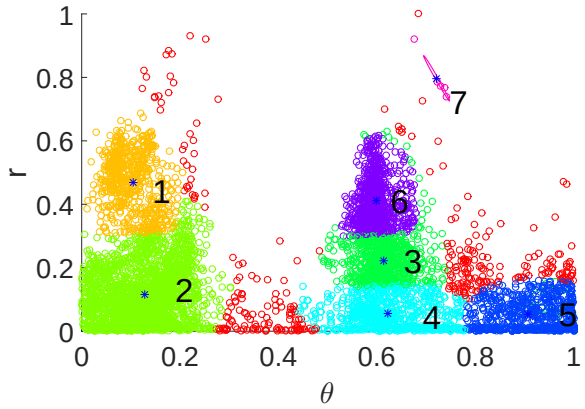
- Cluster **C5W** does not express a unified characteristic of pedestrian behaviour. First, it contains the records that correspond to busy pedestrian activities at 11am and 5pm. However, it also groups all the records collected at the early hour at 6am, where the area is usually vacant with few pedestrians.

- Cluster **C1W** mostly characterizes the pedestrian distribution during late night hours (from 10pm to midnight, then 2am). It is not ideal to see that this cluster also groups two records at 7pm, which suggests that the pedestrian activities of these days are equally low as the midnight level, which contrasts with our understanding of the pedestrian activities of Melbourne. Without concrete evidence, we can only assume that these low activities are anomalous (due to weather, road closure, etc.) but manage to appear in the cluster due to our low threshold of 30% to be included in this table. This highlights one limitation of this analysis of one week's data, where clustering only a small dataset can lead to less robust results.

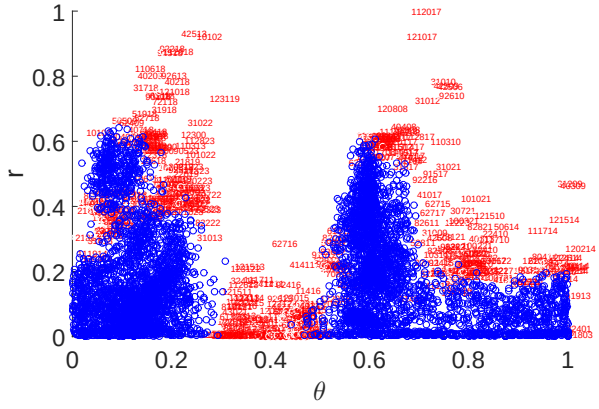
Next, we present our clustering analysis using HyCARCE on one year's worth of data. The results are shown in Table 3.4 and summarized below.



(a)



(b)



(c)

Fig. 3.4 HyCARCE and Ensemble Switching applied on different datasets of pedestrian counting data. (a) HyCARCE on one week's data, (b) HyCARCE on one year's data, (c) Ensemble Switching on one year's data. (θ, r) are the polar coordinates. The notations $*$ mark the centers of the ellipsoids. The texts in (c) indicate the timestamp of the outliers in *mmddhh* (or *mddhh*) format.

- Cluster **C1Y** characterizes the pedestrian distributions during morning peak (9am) and afternoon peak (6pm-7pm).
- Cluster **C2Y** characterizes the pedestrian distributions at the pre-morning peak (7am) and morning peak (8am) period. Similarly to the clustering result of one week's data, the distributions at 8am and 9am are distinguished into two different clusters.
- Cluster **C3Y** and cluster **C4W** corresponds to the low pedestrian activities at 6am and 4pm, and 5am respectively. We highlight that cluster **C5Y** expresses a strong signal of an early hour cluster, which contains records collected at 5am of 253 (out of 261) weekdays of the year 2014.
- Cluster **C6Y** is large and has less clearly defined characteristics. This cluster corresponds to the distribution model spanning from 8pm to 2am the next day. It also packs the distribution at 2pm and 7pm into the same group. These two groups, when interpreted separately, correspond to our understanding of pedestrian activities of Melbourne to some extent: the period 8pm-2am features the continuous activities of pedestrians from the quieter time of the evening to past midnight, while the distribution at 2pm and 7pm characterizes the activities of pedestrians around a major train station in the CBD after lunch and after dinner. However, it is not trivial and ideally not correct to combine them into the same group since the former characterizes low pedestrian activities while the latter features high pedestrian activities. In addition, these two hours do not show unified characteristics of the pedestrian distribution from 8pm to 2am. Besides, we observe that the granularity of detail is low in this cluster. It is arguable that pedestrian activities are different in the evening (8pm-9pm) to late evening (10-midnight) to after midnight (1am-2am). Grouping them all into the same cluster makes the results less granular when compared to the clustering results on one week's worth of data, which managed to distinguish the distributions of the period 8-9pm (**C9W**) from 10pm-midnight (**C1W**), 1am (**C8W**) and 2am (**C1W** and **C2W**) periods.
- **C7Y** contains the observations of 5 different hours of 5 different days spread throughout the year. **C5Y** also contains random hours of all 5 different days of the week, none of the hours have a frequency higher than our threshold of 75% to be considered reliable. Although these two clusters are formed, we are unable to conclude if they exhibit any commonly observed patterns of the pedestrian activities.

Overall, the quality of clustering results has deteriorated. Many values are found to be clustered into the wrong group and the distinguished characteristic of each cluster as found in

Table 3.5 Labels of hours of the day.

Description	Hours	Label
Early morning	6am, 7am	L1
Morning peak	8am, 9am	L2
Afternoon peak	5pm, 6pm	
Lunch time	11am-1pm	L3
Quieter hours between peaks	10am, 2pm-4pm	L4
Evening time	7pm-9pm	L5
Late evening time	10pm-midnight	L6
Midnight to early morning	1am-5am	L7

the previous experiment is no longer observed. This is also reflected in the absence of more hours of the day when compared with the previous result, including 11am-1pm and 5pm. The increase in the volume of data results in a higher data density, which erodes the boundary between clusters. Moreover, the states between different weeks vary intermittently, causing superpositions in the data distribution. This phenomenon intensifies when a long period's worth of data is being analyzed. As an example, the pedestrian activities in a summer week are different from those in a winter week, hence considering these two weeks using a single, static clustering model might overlook the characteristics of each week.

Quantitative evaluation

We then quantify the meaningfulness of our clusters. According to our definition, a meaningful cluster expresses distinguished characteristics of the pedestrian distributions at a certain time of the day. To this end, by assigning a label to each hour of the day and aggregating the predicted characteristics for each cluster, we can measure the meaningfulness of each cluster through its capability of grouping together the hours having the same characteristics of pedestrian distribution. This is quantified using the confusion matrices, precision, recall and F1-score as we present next.

We use our understanding of the pedestrian activities at our locations of interest (Flinders Street-Elizabeth Street and Prince Bridge) to propose 7 different profiles of distributions throughout the day. The descriptions and labels are summarized in Table 3.5. Note that we group both morning peak and afternoon peak to the same group with label L2 since the numbers of pedestrians around this train station area at these hours are mainly dominated by office workers in the CBD, and hence do not vary much between the starting and ending of working hours.

The aggregated predicted characteristic of each cluster is derived by a popularity vote of the labels of its component. Table 3.6 shows an illustrative example on how this was applied

Table 3.6 Illustrative example of deriving the predicted characteristic of clusters. Only the hours whose frequencies are higher than 30% of the maximum frequency are shown in order to keep the table concise, e.g., for one year's data, an hour needs to be present at least 79 times (0.3×261 weekdays) to be shown.

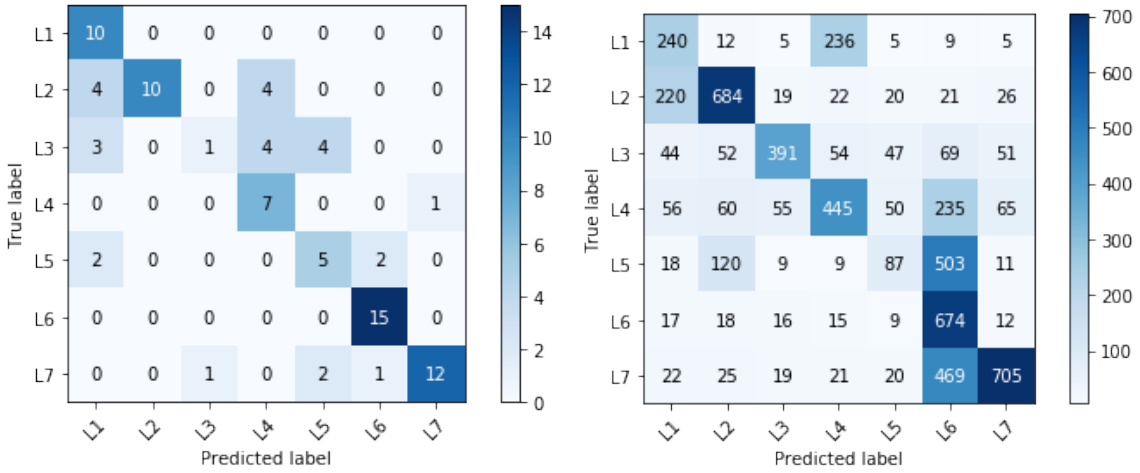
Clusters	(Hour of day, Instance count)	(Label, Instance count)	Predicted characteristics of clusters
C1	(22,5), (23,5), (0,5), (2,2), (19,2)	(L6,5), (L6,5), (L6,5), (L7,2), (L5,2)	L6
C2	(2,2)	(L7,2)	L7
C3	(5,5)	(L7,5)	L7
C4	(7,5)	(L1,5)	L1
C5	(6,5), (17,4), (11,3)	(L1,5), (L2,4), (L3,3)	L1
C6	(8,4), (12,4), (16,5)	(L2,4), (L3,4), (L4,5)	L4
C7	(9,5), (18,5)	(L2,5), (L2,5)	L2
C8	(1,4)	(L7,4)	L7
C9	(13,4), (20,2), (21,3)	(L3,4), (L5,2), (L5,3)	L3

to the clustering result of one week's data. Specifically, cluster **C1W** contains records of 10pm, 11pm, and midnight that correspond to label L6, and records of 2am and 7pm that correspond to label L7 and L5 respectively. Thus, **C1W** contains 15 hours that are associated with label L6, two hours that are associated with label L7, and two hours that correspond to label L5. By using the most frequent label, the predicted label of **C1W** is L6 (late evening time). Subsequently, the *predicted labels* of all instances of this cluster are L6 while the *true labels* consist of L5, L6 and L7 as indicated in the third column of Table 3.6.

We apply the same technique to the clustering result of one year's data then compute and present the confusion matrices as well as the metrics in Figure 3.5, Table 3.7 and Table 3.8. Note that:

- Only the data points that are assigned to a cluster (not an outlier) are included in the metric computation, hence the total count of instances in the confusion matrix does not necessarily add up to the volume of the input.
- We only show in Tables 3.3 and 3.4 the hours whose frequencies are high enough to keep the table concise. Therefore the sum of all instance counts in this table is less than the sum of instances in the confusion matrix.

The results show that HyCARCE clustering results can predict the characteristics of pedestrian distributions, reflected in the high values on the diagonals of the confusion matrices. Here, a high number of correctly predicted labels indicates that the majority of the components of a cluster agree with the labels assigned by the popularity vote. Intuitively, a



(a) (b)

Fig. 3.5 Confusion matrices of predicting characteristics of pedestrian activities for (a) one week's data, (b) one year's data.

Table 3.7 Precision, recall, and F1 score of predicting characteristics of pedestrian activities.

	One week's data	One year's data
Precision	0.731	0.593
Recall	0.682	0.538
F1 score	0.656	0.534

characteristic of pedestrian activities (Table 3.5) is agreed by the majority of the data points of a cluster.

For both results on one week's data and one year's data, HyCARCE can find the clusters that correspond to activities during the period of midnight-5am (label L7) with high precision. This can be explained by the consistently very low activities of pedestrians during these hours due to shop closures and lack of late night public transport. However, it can also be observed that HyCARCE on one year's data does not have high recall (0.55), and many of the records of this group are wrongly put into the cluster of late evening time (label L6). This problem also persists for other periods of the evening: Figure 3.5b shows that multiple records are clustered into a cluster associated with late evening time while their true labels indicate that they should either belong to the period of midnight-5am (469 L7) or earlier evening time (503 L5). This reinforces our argument above about details being lost in cluster **C6Y** in Table 3.4.

In both cases, the algorithm works quite well in identifying the clusters that correspond with peak hours (label L2). While both experiments give high precision and F1 score, the recall when clustering one week's data is slightly lower. Figure 3.5a shows that some of the

Table 3.8 Metrics of predicting characteristics of pedestrian activities by labels.

Labels	Precision		Recall		F1 Score	
	One week	One year	One week	One year	One week	One year
L1	0.526	0.389	1	0.469	0.690	0.425
L2	1	0.704	0.556	0.676	0.714	0.690
L3	0.5	0.761	0.083	0.552	0.143	0.640
L4	0.467	0.555	0.875	0.461	0.609	0.503
L5	0.455	0.366	0.556	0.115	0.5	0.175
L6	0.833	0.340	1	0.886	0.909	0.492
L7	1	0.806	0.750	0.550	0.828	0.654

records of this group are wrongly put in the clusters that correspond to the period of early morning (label L2), or quieter periods between peaks (label L4). Note that the confusion matrix as well as the performance metrics of clustering one week's data might have been affected due to the low volume input. The result might be more reliable if we run HyCARCE on multiple one-week subsets of data and compute the average and variances of precisions and recalls. However, this is not covered in these experiments.

We subsequently use the clustering result of HyCARCE to find anomalies. According to our problem statement, data points that do not belong to any cluster do not correspond to any pedestrian activities that are commonly observed, and hence are regarded as anomalies. The evaluation of anomaly detection using static HyCARCE clustering is discussed in greater detail in comparison with Ensemble Switching Model in the next section.

3.3.5 Evaluation of Anomaly Detection by Ensemble Switching Model and HyCARCE

The set of anomalies produced by HyCARCE is defined as the set of data points that do not belong to any cluster.

Subsequently, we apply the Ensemble Switching Model on one year's worth of data with the window size w of 5 weekdays, and the grid size $g = 0.08$. The two sets of anomalies are then evaluated and compared with each other. To compare the performance between the two methods, we propose a set of ground truth anomalies that are composed of 15 major public holidays and events in the City of Melbourne [2]. The events and ground truth are summarized in Table 3.9. Due to the limited size of the ground truth in this experiment, both methods are evaluated and compared by their recall (and not precision) in detecting anomalous events. The methodology on how these events are selected, what the expected changes are, and what a correct anomaly detection is, are discussed next.

Table 3.9 Public holidays and events selected as ground truth for anomaly detection.

Events	Date	Ground truth hours	Expected change
New Year's Day	Wed, 1 January 2014	1am, 2am	Increased
Lunar New Year	Fri, 31 January 2014	7pm-midnight	Increased
Australian Open	Fri, 24 January 2014	6-10pm	Increased
Australia Day	Mon, 27 January 2014	7am-9am, 5pm-7pm	Decreased
Labour Day (Moomba parade)	Mon, 10 March 2014	9am-3pm	Increased
Formula One	Fri, 14 March 2014	5pm-9pm	Increased
Good Friday	Fri, 18 April 2014	7am-9am, 5pm-7pm	Decreased
Easter Monday	Mon, 21 April 2014	7am-9am, 5pm-7pm	Decreased
ANZAC Day	Fri, 25 April 2014	7am-9am, 5pm-7pm	Decreased
Queen's Birthday	Mon, 9 June 2014	7am-9am, 5pm-7pm	Decreased
Grand Final Parade	Fri, 26 September 2014	11am-3pm	Increased
Melbourne Cup	Tue, 4 November 2014	7am-9am, 1pm-3pm	7am-9am: Decreased 1pm-3pm: Increased
Christmas Festival	Wed, 24 December	6pm-11pm	Increased
Christmas Day	Thu, 25 December 2014	7am-9am, 5pm-7pm	Decreased
Boxing Day	Fri, 26 December 2014	7am-9am, 10pm-6pm	7am-9am: Decreased 10pm-6pm: Increased

We select 10 public holidays, and 5 festival and sporting events of the City of Melbourne in 2014 to use as the reference set of anomalous events to evaluate and compare the performance of the two methods. Due to the limitation of this experiment that considers only weekday data, we only select the events that fall on weekdays. We then checked the schedule and used our best understanding of the festival and sporting events to label the hours that are affected and use these hours as the ground truth. For example, the Australian Open semi final took place in the evening of 24^{th} January 2014, so only the pedestrian activities from 6pm to 10pm that date are considered anomalous. If the algorithm detects other hours of the day as anomalies, they are still considered false positives. Besides, if the reference is a public holiday that does not associate with any event, we used 7am-9am and 5pm-7pm as ground truth. These two hours correspond to the morning peak and afternoon peak and we believe the pedestrian activities around our locations of investigation (a main train station) differ significantly from those on a normal weekday. We also label the expected direction of change in pedestrian counts. For example, pedestrian counts are expected to decrease during the morning peak of Australia Day and increase on Boxing Day (more pedestrians around train stations and shops in the biggest shopping day of the year). Table 3.9 summarizes our ground truth.

Table 3.10 Confusion matrix of anomalies detected by ESM and HyCARCE

	Ensemble Switching Model		HyCARCE	
	Predicted: Anomaly	Predicted: Non-Anomaly	Predicted: Anomaly	Predicted: Non-Anomaly
Actual: Anomaly	13	2	10	5
Actual: Non-Anomaly	363	5886	257	5992

If the algorithm can detect at least one hour of the ground truth, we subsequently compared the actual count value with the average pedestrian count *at the same hour* of other days of that week. If the comparison matches with what is expected in Table 3.9, we considered that the algorithm successfully detects that anomaly. Each method was assessed in terms of whether it can detect the anomalies, and if so, whether the results match closely with the actual schedule of the events. The confusion matrix is presented in Table 3.10 and the result evaluation is shown in Table 3.11.

Table 3.11 Anomalies detected during major events from Jan to Mar 2014.

	Detected anomalies	
Events	Ensemble Switching Model	HyCARCE
New Year Day	1-3am	1-2am
Lunar New Year	8-9pm	not detected
Australia open	9pm	not detected
Australia Day	7am-9am	7am-9am
Moomba parade	9am-1pm, 9-10pm	12pm, 9-10pm
Formula one	5-8pm	11pm (false positive)
Good Friday	7am-9am, 5pm-8pm	7am-9am
Easter Monday	7am-9am, 5pm-6pm	7am-9am, 5pm-7pm
ANZAC Day	not detected	not detected
Queen's Birthday	7am-9am, 5pm-7pm	7am-9am, 5pm-7pm
AFL Grand Final Parade	11am-3pm	12pm-3pm
Melbourne Cup	not detected	not detected
Christmas Festival	10pm-11pm	8pm
Christmas Day	7am-9am, 5pm-6pm	7am-9am, 6pm-7pm
Boxing Day	10am-2pm, 3pm-5pm	10am-2pm, 3pm

Out of 15 events, ESM correctly detects 13 anomalies while HyCARCE detects 10 anomalies. In addition, spot checking shows that the output of ESM provides better coverage and matches the schedule of the events more accurately. For example, the Moomba parade took place at 11am on 10^{th} Mar 2014, and it is likely that a higher than normal number

of pedestrians would be observed before, during and after the event; hence an anomalous distribution of pedestrians from 9am to 1pm as returned by ESM are more accurate compared to the HyCARCE result. The same observation about better coverage also applies to the AFL Grand Final Parade, Good Friday, and Christmas Festival. Although HyCARCE also detects some of the unusual events, many valid anomalies were not detected. Note that HyCARCE classified the pedestrian distribution at 11pm on the day of the Formula One event as an anomaly. This is a false positive as the event would have finished 3 hours earlier.

Using our small set of ground truth, ESM has a higher true positive rate (recall) than HyCARCE. At the same time, Table 3.10 also shows it has a higher number of false positives (363 vs 257), which is not conclusively worse due to the settings of this experiment. We simply labeled any records that fall outside of our 15 public holidays and events as non-anomalies. In reality, many things can result in anomalous pedestrian activities, such as weather, transportation disruption, street performers, infrastructure maintenance, etc., for which it is challenging to derive a ground truth. For this reason, the fact that ESM predicted more anomalies (and subsequently more false positives) does not imply its inferior performance. The comparison on false positives is more reliable if we have a larger set of labeled anomalies. For this reason, we only use recall and not precision to compare the two algorithms in this experiment.

In addition to the comparison of the confusion matrices of ESM and HyCARCE in Table 3.10 above, we also made the following observations when spot checking the results of ESM:

- The method is more robust to false negatives because each data point is assessed for being an anomaly using not only its own window of data but also with the support of cluster models of other relevant windows. A good example of a false negative is the pedestrian count at 11pm on 5^{th} Dec 2014. In its window, this observation was grouped in the cluster of 6pm because their values share some proximity. However this erroneous clustering is corrected when the point is assessed with other windows and consequently classified as an anomaly. Checking with the city event calendar, there was a public movie screening organised right next to Princes Bridge until 11pm on the same day, which can explain and support this decision.

- By splitting the input into multiple windows, the method is capable of handling high density data and superpositions of different data distributions. Figure 3.4c shows all the anomalies in red. A quick examination shows that although these values are in close proximity, they belong to random hours of different days throughout the year, and do not express any trends or repeating patterns in pedestrian activities.

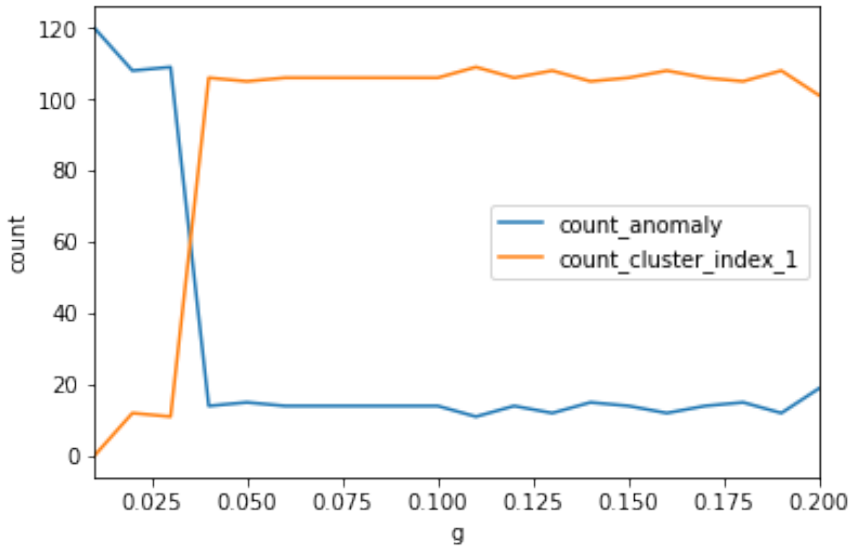
3.3.6 HyCARCE and ESM with Higher Dimensional Data

Having observed that HyCARCE can cluster one year of data on two dimensions, and that both HyCARCE and ESM can detect anomalies, we aim to scale the approach to higher dimensional data, to include sensor data from more locations. To this end, we first attempt to perform HyCARCE clustering on 3-dimensional data that is composed of sensor data collected from Flinders St-Elizabeth St, Princes Bridge and Melbourne Central. We also perform the same data transformation to the input data that computes the differences between successive counts. However, we cannot apply the polar transformation since it is only applicable for two-dimensional data. We also apply HyCARCE on one week's and one year's worth of data, and perform the search for optimal value of g in the range of $[0.01, 0.2]$.

In either case of clustering one week's data or one year's data, a common pattern of the results emerges showing that HyCARCE can only find at most one big cluster that covers almost every data point and discards the rest as anomalies (i.e., points that do not belong to any clusters). Figure 3.6a suggests that with $g \leq 0.03$, HyCARCE can only find one small cluster that contains 11 points and the rest do not belong to any cluster. When $g > 0.03$, HyCARCE finds one single big cluster that covers 106 (out of 120) data points. Note that we have 120 data points in one week's data, and the graph also shows that the sum of `count_anomaly` and `count_cluster_index_1` is equal to 120. The same pattern can also be observed in Figure 3.6b: although the changes in cluster sizes are not as steep, only one cluster is found. When we analyze the components of the single big cluster in each case, we observe that it contains almost every hour of the day and does not express any distinguished characteristics of the type of pedestrian activities. To this end, we conclude that the HyCARCE results when clustering this set of 3-dimensional data are not meaningful.

For the completeness of the discussion, we still include the result of ESM on this 3-dimensional dataset. For all values of g in the range $[0.01, 0.2]$, we always find a fixed set of 684 anomalies. Therefore, having ensemble clustering results on sliding windows does not have any effect on its ability to detect anomalies. This is probably because performing HyCARCE on each window does not yield good clustering (e.g., only one big cluster as discussed above).

The unsuccessful result could be accounted for by the difference in the Euclidean and polar coordinates. Nevertheless, being forced to apply a specific transformation for the approach to work is a limitation that prevents the approach to be scaled to higher dimensional data. We believe a more generic reason that makes the problem more challenging in this experiment is due to the need for local feature relevance as described in Section 2.2.1. Pedestrian counts are correlated at only some locations but not all, i.e., the clusters are formed in subspaces rather than the full data space. In this case, the pedestrian counts at



(a)

(b)

Fig. 3.6 Cluster sizes and counts of anomalies in the clustering results of (a) one week's data, (b) one year's data. Note that *count_cluster_index_*1 denotes the size of cluster C1.

Flinders Street Station and Princes Bridge have strong correlations as they are located near each other. However, pedestrian counts at more distant locations such as Melbourne Central might not always be relevant. When considering each pair of dimensions separately, their correlation can be revealed and hence clusters can be formed. However, taking into account all these locations when measuring the similarity between different time points might hinder the latent correlation. For example, the decreasing pedestrian activities at Flinders Station and

Princes Bridge between 7am-8am can be less evident if the increasing pedestrian activities at Melbourne Central are included and all three locations are considered simultaneously. In addition, at different times of the day, the pedestrian activities at a location may be correlated to those of different locations.

3.3.7 Summary of the Findings of HyCARCE and ESM

We have modelled the pedestrian load profile at two locations of the City of Melbourne and use the model to extract anomalies. We have first used HyCARCE, which is a static clustering approach, to cluster pedestrian distributions at different hours of the day into groups and subsequently mapped these groups into types of pedestrian activities of the day, such as morning peak, afternoon peak, and lunch time. Based on our interpretation of the clusters of one year's data, as well as the metrics to measure cluster accuracy, we conclude that the pedestrian distribution can be clustered into some meaningful profiles characterizing major activities throughout the day, especially the morning and afternoon peak, lunch time, late evening time, and midnight to next morning period. The results, however, do not cluster well the off-peak periods, or the early evening period. We then used the clustering result of HyCARCE to detect anomalies.

We then evaluated Ensemble Switching Model (ESM), which is an anomaly detection algorithm that can work with nonstationarity in dynamic systems. By comparing the effectiveness of ESM and HyCARCE on detecting anomalies with our proposed ground truth labels, we have shown that ESM outperforms HyCARCE. Note that the set of ground truth that we use only contains 15 events, hence the results can be more reliable and insightful if we can have a larger set of ground truth labels.

This, however, is our preliminary work and uses only two dimensional data. When we attempt to include more locations in the experiments (i.e., the number of dimensions increases to more than two), the clustering results do not show any clear grouping of hours that can be mapped to major activities of the day. In order to make the model practical and more useful, it needs to be able to work with multi-dimensional data and reveal the patterns across multiple locations. The next section discusses our subsequent work with higher dimensional data to construct the distribution profile of pedestrians across multiple locations in the CBD for the purpose of anomaly detection.

3.4 Frequent Pattern-Based Approach

Expanding on the work in the previous section, we aim to model the pedestrian distributions across the city and consider observations at multiple locations simultaneously (rather than just two). Our initial attempts using HyCARCE and the Ensemble Switching Model found that these two methods have limitations in handling higher dimensional data due to local feature relevance as described in Section 2.2.1.

This section presents the research challenges we encounter when analyzing multi dimensional pedestrian data, followed by our methodology to model and cluster such data. Specifically, we develop an algorithm that (1) can handle multi dimensional data with attributes that share different degrees of correlation and (2) can cope with a non-stationary system that switches between different states over time.

3.4.1 Problem Statement

Let X denote the set of N observations from time t_1 to t_N: $X = \{x_i : i = 1..N\}$ where $x_i \in \mathbb{R}^d$ is a vector of d dimensions corresponding to the pedestrian counts observed at d different locations at time t_i. We also denote h_i as the hourly component of time t_i, and $X_{h_j} = \{x_{h_j}\} \subset X$ as the set of pedestrian counts in X that are observed at hour j in different days. For example, $\{x_{h_{17}}\}$ denotes the pedestrian counts that are observed at 5pm.

First, we aim to model the normal behaviour of X as the set of correlations of pedestrian distributions at different locations during the same period $\{h_j\}$ of the day (i.e., a period can span multiple hours). We adopt the technique of frequent pattern mining to learn the associations of pedestrian distributions at different locations. To this end, we consider each observation x_i as a transaction and the pedestrian count category at *each location* during time t_i as *an item* of that transaction. Our algorithm finds the set of patterns $F_{\{h_j\}}$ that express the normal patterns of pedestrians during the hours $\{j\}$ of the day. It is defined as the *frequent patterns* extracted from the input data subset $X_{\{h_j\}}$:

$F_{\{h_j\}} = \{FrequentPattern_{\{h_j\}}\}$, $FrequentPattern_{\{h_j\}}$ is a set of frequent patterns of $X_{\{h_j\}}$.

For example, if we aim to model the normal behaviour of pedestrian during the morning peak then the interpretation of our notations is as follows:

- $\{j\} = \{7,8,9\}$, this set represents the hours of morning peak.
- $X_{\{h_j\}}$ contains all the pedestrian counts that are observed at 7am, 8am and 9am.

- $FrequentPattern_{\{h_j\}}$ represents a pattern that is frequently observed at 7am, 8am and 9am. This pattern is extracted from $X_{\{h_j\}}$.
- $F_{\{h_j\}}$ contains all $FrequentPattern_{\{h_j\}}$, and represents all the patterns of pedestrians that are frequently observed during this period of the day.

Subsequently, we aim to find a set $A_{\{h_j\}}$ containing all the anomalies that are not covered by any frequent patterns during the period $\{h_j\}$ of the day. Note that the scope of the normal behaviours $F_{\{h_j\}}$ as well as the anomalies $A_{\{h_j\}}$ is limited by the period $\{h_j\}$ in this experiment. We elaborate the reasoning and discuss its limitations in the subsequent sections.

By evaluating the set of patterns $F_{\{h_j\}}$ at different periods of the day, we seek to understand the correlations of pedestrian distributions at different locations of the city and use that knowledge to systematically validate and explain the anomalies found in set $A_{\{h_j\}}$.

3.4.2 Challenges

In order to describe the distribution of pedestrians across the city, we consider observations at multiple locations in the city, resulting in high dimensional input data. According to Kriegel et al. [128], clustering of high dimensional data poses multiple challenges for clustering algorithms because traditional similarity measures become sensitive to noise. Challenges also arise from local feature relevance in high dimensional data, which entails that clusters might exist in different subspaces but not in the full data space being considered. For example, the pedestrian counts at Flinders Street Station, Princes Bridge and the Arts Center in Melbourne have strong correlations as they are located near each other. However, the pedestrian counts at more distant locations such as the Queen Victoria Market or China Town, might not be relevant, and might possibly prevent the formation of clusters. Besides the high dimensionality, the data used in this study is not evenly distributed, with a large proportion of the observations being skewed towards the lower values, which makes clustering even more difficult. The challenge of local feature relevance is detailed in Section 2.2.1.

Another challenge in modelling the data comes from the long time scale of the data being collected, which might cause the model of normal behaviour to vary within different time frames as pedestrian behaviours may change intermittently throughout the year. This phenomenon is discussed in greater detail in Section 3.3 where we present how we use the Ensemble Switching Model to tackle this challenge for low dimensional data.

Moreover, modelling pedestrian activities is more complicated than the task of pure frequent itemset mining. One of the major challenges is that the output frequent patterns that form the model need to be reliable, meaningful and representative of the dataset. We need to ensure that the set of frequent patterns are sufficiently representative of both the observations

and the locations being analyzed. Note that we consider the pedestrian counts as a transaction database, in which each transaction contains the counts of pedestrians at different locations at a certain hour, i.e., an observation of the pedestrian distribution. Hence, each frequent pattern being mined represents a distribution of pedestrians that are frequently observed; the items of that frequent pattern indicate the locations where pedestrian counts are correlated. A result with multiple short and simple patterns will just express the fragmented relationships between small groups of locations, but not the relationships between a larger set of locations in the city. On the other hand, overly complicated patterns cover only a small portion of the observations since not many data points can be matched with these complicated patterns. In addition, these patterns divide the input into multiple groups with excessive granularity and make the model less intuitive.

As a result, the following constraints are imposed:

Constraint 1: The support σ of each frequent pattern $FrequentPattern_{\{h_j\}}$ needs to be high.

Constraint 2: The combination of frequent patterns in $F_{\{h_j\}}$ needs to cover a significant proportion of observations in the input dataset. This constraint can also be regarded as the *observation coverage*, i.e., the total number of observations that belong to at least a frequent pattern. While the support σ in **Constraint 1** *is local to each frequent pattern*, a high value of observation coverage ensures that the model does not miss any common patterns in the entire dataset being analyzed.

Constraint 3: The union of all items (locations) in the frequent sets need to cover a sufficient proportion of locations being investigated. This requirement can be regarded as the *location coverage*. It prevents constraints (1) and (2) from being overwhelmed by small frequent patterns. For example, we observed from our experiments that it is easy to find many small frequent 2-itemsets with very high support (near 100%). To that end, if a high `min_sup` (minimum support threshold) is selected, constraints (1) and (2) are easily satisfied by these 2-itemsets. However, in these cases, the model is dominated by patterns of only a few locations and many other locations are left out. To that end, the model only covers a small portion of locations in the city and does not offer any useful insights on the locations that are left out.

3.4.3 Methodology

This section describes our algorithm to model the normal pedestrian distribution and to detect anomalous behaviors. The algorithm comprises two phases - data normalization and modelling normal behavior.

Data Normalization

The count values collected by the sensors offer few insights on the correlations between pedestrian distributions at different locations. For example, a count of 1500 pedestrians is unusually high for the Queen Victoria Market. However, at Flinders Street Station, it is considered low during most times of the year. Data normalization aims to better reveal the relationships of the count values at different locations in different time frames. Our approach in modelling the normal behaviour of pedestrians adopts the Apriori algorithm [38], which requires categorical input data as transactions of items. For this reason, the same normalization technique that we used in Section 3.3.3 cannot be reused here.

Since the system can change between different states intermittently, comparing the counts of the same sensor at different times can lead to a misleading estimate of the true distribution. We divide the stream of data into windows of size w. In each window, the maximum count at each location is chosen as the standard measure (100%) and all the remaining values are classified into buckets of `LOW`, `MEDIUM`, `HIGH` in proportion to their corresponding measure using the scale below:

$[0\%, 30\%) \rightarrow$ `LOW (L)`
$[30\%, 70\%) \rightarrow$ `MEDIUM (M)`
$[70\%, 100\%] \rightarrow$ `HIGH (H)`

By dividing the data into small windows before classifying the pedestrian counts into buckets, it is observed that the classification reflects the correlations more accurately because it is assessed locally in a short time frame and is not biased by factors that change over time, such as weather or holiday periods. This step of normalization also makes the algorithm more robust when the system switches between states. In practice, the window size w should be selected such that each window covers a cycle of events that are sufficiently small not to be affected by the seasonality of data. For example, a window size of three or six months is probably too long because the pedestrian activities can change intermittently during the period, e.g., hot weather vs rainy weather. In this experiment, we tried with both window sizes of one week and two weeks. We obtain better results with a window size of two weeks and only report that result in the next section.

As an illustration, the data normalization process is applied on the data at Flinders Street Station and Southern Cross Station during the morning peak (i.e., 7am-9am) between 15^{th} Dec and 31^{st} Dec 2014. The process is shown in Table 3.12. This example also demonstrates the importance of the normalization to express the true nature of the pedestrian distribution at each station so that their correlations can be analyzed more accurately. As shown in Table 3.12, the counts of 769 pedestrians at Flinders Street Station and 773 at Southern Cross

Station are very similar, however they represent two different distributions at two locations: *LOW* at the former and *MEDIUM* at the latter location.

Table 3.12 Example of data normalization of pedestrian counts at Flinders Street and Southern Cross Station for the data window 15th Dec - 31st Dec 2014. The maximum value of the window is highlighted in grey.

Dates (Dec)	15^{th}	16^{th}	17^{th}	18^{th}	19^{th}	20^{th}	21^{st}	22^{nd}	23^{rd}	24^{th}	...	28^{th}	29^{th}	30^{th}	31^{st}
Count values	142	1089	2338	2280	1423	1068	219	769	849	1625	...	1829	2040	2914	2446
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓		↓	↓	↓	↓
Percentage of max value	4.8	37.3	80.2	78.2	48.8	36.6	7.5	26.3	29.1	55.7	...	62.7	70.0	100	83.9
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓		↓	↓	↓	↓
Bucket	L	M	H	H	M	M	L	L	L	M	...	H	H	H	H

(a) Flinders Street Station

Dates (Dec)	15^{th}	16^{th}	17^{th}	18^{th}	19^{th}	20^{th}	21^{st}	22^{nd}	23^{rd}	24^{th}	...	28^{th}	29^{th}	30^{th}	31^{st}
Count values	7	117	1419	1987	2404	2235	773	349	1115	1802	...	377	337	479	441
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓		↓	↓	↓	↓
Percentage of max value	0.3	4.9	59.0	82.7	100	92.9	32.2	14.5	46.3	74.9	...	15.7	14.0	19.9	18.3
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓		↓	↓	↓	↓
Bucket	L	L	M	H	H	H	M	L	M	H	...	L	L	L	L

(b) Southern Cross Station

The categorized data is then binarized [101] before we apply frequent itemset mining in the next phase to model the normal behaviour. In this step, each location l is then assigned with three attributes l_{LOW}, l_{MED} and l_{HIGH} which express the presence or absence of that type of count range at the location.

Normal Behaviour Modelling and Anomaly Detection

The Apriori algorithm [38] is used as the underlying technique to mine frequent itemsets from the input. The main challenges of the modelling process lie in selecting an appropriate support threshold that balances the coverage of the observations, the coverage of locations and the reliability of the frequent itemsets as mentioned in the problem statement. A support threshold that is too low increases the chance that multiple locations are included into the frequent sets and more observations are covered, but may result in frequent itemsets that are not reliable. On the other hand, a support threshold that is too high guarantees that the frequent sets are strong, but at the same time might exclude more locations from the frequent sets. The algorithm used to achieve an appropriate mix of frequent itemsets is described in Algorithm 1.

The algorithm attempts to find the highest support threshold for which the frequent itemsets achieve a good coverage of the locations as well as represent a majority of the observations. The algorithm is initialized with the highest support threshold `min_sup`. Then at each iteration, `min_sup` is decreased and the corresponding frequent itemsets are generated and tested to see whether they provide a good model. The algorithm stops when the first set of frequent itemsets that satisfies all the requirements above is found, or when `min_sup` reaches the lower bound threshold. As described in **Constraint 2**, `min_sup` is only local to each frequent pattern while a constraint on observation coverage guarantees that the ensemble of frequent patterns reflect the majority of observations. In this study, we set the threshold of observation to be covered $oCoverage = 80\%$, the threshold of locations to be covered $lCoverage = 70\%$ and the support decrement step to 5%. *Any records that do not match any of the frequent itemsets are considered as anomalies*.

Next, we present an illustrative example of the application of the algorithm.

Algorithm 1: Modelling normal behaviour

Input : $O_{\{h_j\}}$: set of N normalized observations at hours $\{j\}$ of the day
oCoverage: threshold of observations to be covered
lCoverage: threshold of locations to be covered
sThreshold: threshold of minimum support

Output : $F_{\{h_j\}}$: set of all frequent itemsets at hours $\{j\}$ that satisfy the constraints

```
1  min_sup := 1;
2  found := false;
3  while (not found and min_sup > sThreshold) do
4      F_{h_j} = apriori(O_{h_j}, min_sup);
5      // remove the 2-itemsets from F_{h_j}
6      F_{h_j} :=remove2ItemSet(F_{h_j});
7      // test the coverage of the model
8      cl := findLocationCoverage(F_{h_j});
9      co := findObservationCoverage(F_{h_j});
10     if (cl > lCoverage and co > oCoverage) then
11         found := true;
12     end
13     else min_sup := min_sup − 0.05 ;
14 end
15 return F_{h_j}
```

An Illustrative Example

The input is a 4-dimensional dataset collected by the sensors at Flinders Street Station (FL), Southern Cross Station (SC), Princes Bridge (PB) and the Arts Center (AC) during morning peaks (7am-9am) from 1^{st} Jun to 31^{st} Dec 2014.

The frequent patterns that are mined as well as their corresponding coverage on the observations and on the locations in each iteration are shown in Table 3.13.

Initially, `min_sup` is set to 100% and no frequent itemsets can be found. The minimum support threshold is then decremented at a step of 5% in each iteration. In the next six iterations, where `min_sup` is decreased from 100% to 75%, only one frequent itemset can be mined from the data, but it contains only two items and is discarded anyway. In this study we decided to ignore all the 2-itemsets in the output because they create bias when judging how many observations are covered by the frequent itemsets. It was observed from our experiments that in most of the cases, the 2-itemsets have very high support, i.e., they cover a large proportion of the data, which makes **Constraint 2** less meaningful. Including these small sets might lead to a poor model that contains only 2-itemsets. In this case, all three constraints are satisfied but the model only barely contains correlations in terms of pairs of locations, rather than the correlations of pedestrian distributions at multiple locations as a whole.

In the 7th iteration, with `min_sup` decreased to 70%, the output 3-itemset satisfies **Constraint 3** but covers fewer observations than required. In the 8th iteration, the Apriori algorithm finds two 3-itemsets that, in combination, cover 88% of all the observations. Moreover, all locations are covered by this output. All three constraints are met and the algorithm terminates. The itemsets found in the 8th iteration form the model that describes

Iteration	Outputs	Constraint checks
1	$\emptyset$	$oC = 0\%$ $lC = 0\%$
2 3 4 5 6	$\{PB\_LOW, SC\_LOW\}$ (discarded)	$oC = 98\%$ $lC = 50\%$
7	$\{FL\_MED, PB\_LOW, SC\_LOW\}$	$oC = 73\%$ $lC = 75\%$
8	$\{FL\_MED, PB\_LOW, SC\_LOW\}$, $\{PB\_LOW, SC\_LOW, AC\_MED\}$	$oC = 88\%$ $lC = 100\%$

Table 3.13 Illustration of the iterations of Algorithm 1. oC, lC are the coverage levels of the frequent itemsets on the observations and locations respectively.

the normal correlations of pedestrian distributions at Flinders Street Station, Southern Cross Station, Princes Bridge and the Arts Center during the morning peak.

3.4.4 Experiments

We perform experiments on the pedestrian counts at 10 different locations around the City of Melbourne. The locations are highlighted in Figure 3.7. For the full details of the locations, refer to Table A.2 in the Appendix. These monitoring points are located at the major train stations, city hubs, shopping malls or tourist attractions, and are representative of the flows of pedestrians inward, outward and within the CBD area. Specifically, by monitoring the exits of three major train stations in the CBD (Flinders Street station - locations 5 and 7, Southern Cross station - locations 16 and 17, and Melbourne Central station - location 14), we expect to capture the movements of workers in the main office area of the CBD. Furthermore, sensors deployed at the Melbourne Central station and the State Library (location 14), Town Hall (location 21), Princes Bridge (location 4), the Arts Center (location 1) covers Swanston Street, one of the backbones of the CBD of the City of Melbourne that is filled with shops, restaurants, food courts, theatres and tourist attractions. We believe that these are the entry points of pedestrian movements that are directly impacted by any anomalous events in the city.

We focus on two periods of the day: morning peak (7am-9am) and midnight (10pm-midnight) to keep the window small. A pattern is composed of frequently repeated characteristics of the distribution of pedestrians. Since we have only three levels of discretization in this experimental setting, and the distributions of pedestrians change throughout the day, interesting patterns at a certain time can be obscured and considered normal in larger windows. For example, the numbers of pedestrians at Southern Cross station are usually considered `HIGH` in the morning at 8am, so any `LOW` distribution is considered as an interesting event. However, if we carry out the experiment on the whole day in one go, the `LOW` distributions at Southern Cross station at other times of the day (11am, 3pm, evening hours) then form its own frequent pattern and include other interesting `LOW` distributions in this pattern. The anomalies are then undetected. We describe the example above using only one location, however, we believe the observation about interesting events being diluted also applies to multiple locations. The locations selected in this experiment are interconnected and the changes are usually homogeneous, e.g., we see pedestrians at Southern Cross, Flinders Street, and Melbourne Central stations rise and fall together. However, we show that our anomaly detection approach can still detect anomalies that are not homogeneous across all locations being investigated. Nevertheless, having the analysis on separate small windows of the day

Table 3.14 Parameters used for Algorithm 1 to model normal behavior of pedestrians.

sThreshold	rCoverage	lCoverage
40%	80%	80%

is a limitation of this experiment as that will not reveal the similarities between patterns at different times of the day or how they develop over time throughout the day.

We extract the subsets of observations $X_{morning}$ (7-9am) and $X_{midnight}$ (10pm-midnight) and use each input separately for the experiment of pedestrian modelling and anomaly detection.

In the next section we present and discuss the results on six months of data, from 1st Jun to 31st Dec 2014. Similar to the experiments in Section 3.3, we also exclude weekends, resulting in the input of 480 10-dimensional data points. Table 3.14 describes the parameters used in **Algorithm 1** to find the set of frequent itemsets that satisfy all of our three constraints. The window size w for data normalization is two weeks. We conduct two phases of verification on the results of anomalies:

Fig. 3.7 Deployment map of sensors (sensors circled in red are used in this study). Figure reproduced from the website of Pedestrian Counting System of the City of Melbourne [1].

1. *High level verification:* In this phase we empirically compare the anomalies against the model of normal behaviors. The differences between the two sets of events can provide a high level overview of the effectiveness of the algorithm, since it is expected that anomalies will strongly show different attributes compared to the model of normal behaviors.

2. *Concrete verification:* We propose a small set of ground truth labels of anomalous events based on our understanding of pedestrians of the City of Melbourne. We then manually validate whether we can detect these anomalies. The results are first discussed qualitatively and subsequently quantified into metrics.

High Level Verification

We found 21 anomalies spread over 10 different days during the morning peaks and 91 anomalies spread over 60 days during midnight periods.

To highlight the contrast between normal behaviors and anomalous events detected by our algorithm, we first extract the distributions corresponding to data points that are covered by at least one frequent itemset (i.e., it belongs to normal behavior) then count the frequencies of each category of counts (i.e., `LOW`, `MED`, or `HIGH`) at each location. Similarly, we count the frequencies of each category of counts of the anomalies. Next, we use these to compare the differences between the distributions of pedestrians during normal versus anomalous events, and show the comparison for Flinders Street station, Town Hall West, Southern Cross station, and Melbourne Central station during morning peaks in Table 3.15. Table 3.15 shows that, on the aggregated level at 7am, Flinders Street station observes `LOW` distributions 11 times, `MED` distributions 142 times, and does not observe `HIGH` distributions.

Table 3.15 Comparison of categories of counts of pedestrians at Flinders Street and Southern Cross Stations.

		Flinders Street			Town Hall West			Southern Cross			Melbourne Central		
		L	M	H	L	M	H	L	M	H	L	M	H
Normal	7am	11	142	0	153	0	0	23	130	0	153	0	0
	8am	7	86	60	83	70	0	9	75	69	139	14	0
	9am	38	115	0	18	135	0	53	100	0	131	22	0
Anomalies	7am	9	0	0	8	2	0	9	0	0	8	2	0
	8am	7	0	0	6	1	0	8	0	0	8	0	0
	9am	5	0	0	4	0	0	4	0	0	3	0	0

The differences between the two sets of normal and anomalous records are clear. Flinders Street station usually observes `MED` distribution at 7am, then increases towards `MED-HIGH` at 8am and then decreases back to `MED` at 9am. However, the anomalous records show totally different trends where the distributions always remain `LOW`. While anomalies detected at Flinders Street station and Southern Cross station feature a drop in pedestrian traffic compared to normal behavior from 7am to 9am, anomalies observed at Town Hall West and Melbourne Central feature different patterns, which is an increase in pedestrian traffic at 7am (more `MED` distributions at 7am), following a drop in traffic at 8am and 9am.

In order to better analyze the patterns as well as compare the differences between normal and anomalous records at all 10 locations, we display the types of distribution of pedestrians as an intensity image in Figure 3.8. For time t_p of the day, each location is represented by a block of gray tone. The intensity of the gray tone is proportional to the average level of the pedestrian distribution at time t_p in the dataset, which is defined as:

$$P_p = \frac{\alpha_1|LOW| + \alpha_2|MEDIUM| + \alpha_3|HIGH|}{|LOW| + |MEDIUM| + |HIGH|}$$

where $|LOW|$, $|MEDIUM|$, $|HIGH|$ are respectively the numbers of `LOW`, `MEDIUM` and `HIGH` distributions and α_1, α_2, α_3 are the corresponding weights to distinguish the contributions of each type of distribution to the intensity image. We set $\alpha_1 = 0.15$, $\alpha_2 = 0.3$, and $\alpha_3 = 0.55$ in this experiment to give higher weights to `HIGH` and `MED` distributions.

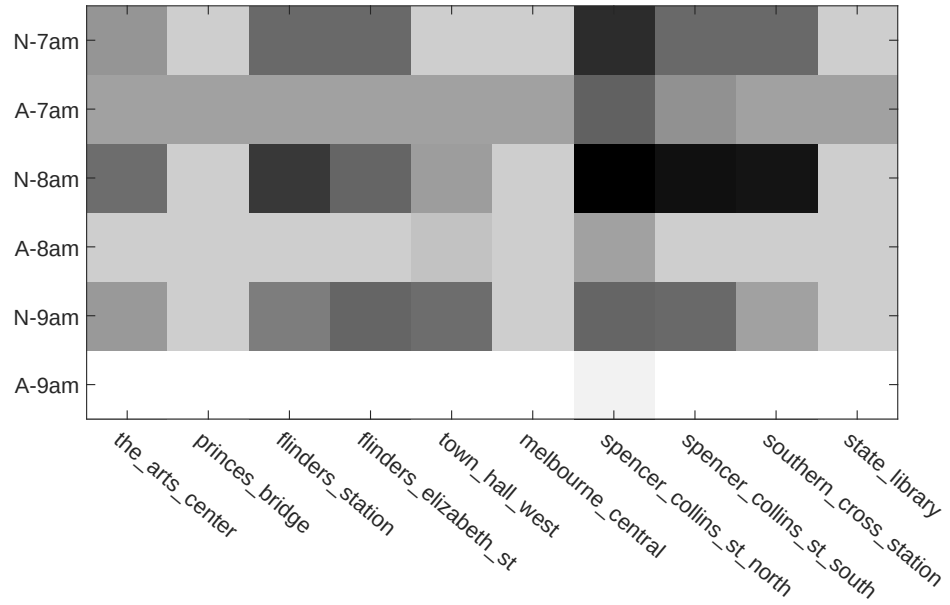


Fig. 3.8 Contrast between normal behaviour and anomalous events at 10 locations during morning peaks. Blocks with darker grey represent higher distributions of pedestrians at the given location and vice versa.

The differences can be observed at almost every location for each time slot. The contrasts between the normal (N-7am, N-8am and N-9am) and anomalous records (A-7am, A-8am and A-9am) shown in Figure 3.8 indicate that the different behaviors are significant. For example, during the anomalous events at Town Hall West, the number of pedestrians is higher than usual at 7am, it then decreases to slightly lower than usual at 8am, and to a much lower level at 9am. At 8am and 9am, most anomalies experience a drop in traffic across all

locations, represented by lighter shades of intensity of A-8am versus N-8am, and A-9am versus N-9am. The drops in pedestrian numbers at 9am between normal and anomalous events are more noticeably significant. On the other hand, anomalies observed at 7am express mixed changes in pedestrian counts across different locations: the numbers of pedestrians increase at most locations along Swanston Street, including Princes Bridge, Town Hall West, Melbourne Central, and State Library (opposite to Melbourne Central) while decrease at major train stations and surrounding areas, such as the Flinders Street station (both exits), the Arts Center, Southern Cross station and its nearby intersection of Spencer Street/Collins Street. This indicates a movement of pedestrians out of train stations that are near to office areas into more recreational areas.

Concrete verification

Ground truth

In this section we select a subset of known events, during which we believe the pedestrian distributions are different from the norm, and use it as ground truth to evaluate the experiment in a more concrete and quantifiable way. Although we already propose a set of ground truth events in Table 3.9, this cannot be fully reused for two reasons: (1) only 6 of the ground truth events are within the period of our analysis (from 1^{st} June to 31^{st} December), and (2) some of these events happen outside of the windows we focus on (7am-9am, and 10pm-midnight). For these reasons, we reuse what is applicable from Table 3.9 and propose two new sets of ground truth events of 10 records each to evaluate our approach during morning peak and evening hours.

As most of the events do not take place in the morning, we select the days in the last weeks of the year of 2014 as anomalies for the 7am-9am period. During these days, the two main universities in the CBD are closed, multiple companies have forced annual leave, in addition to multiple people taking holidays, which leads to a drop in the number of pedestrians during morning peak. However, pedestrian numbers are already low and do not vary much during evening time, even during this year end holiday period; hence they cannot be used as ground truth for anomalies during evening time. We then use the event listing website [2, 9] to select festivals and sporting events during the night time and use them as a ground truth. The ground truth events are described in Table 3.16 and Table 3.17.

The expected change in the tables indicate the aggregated expected changes across all the locations (sum of all pedestrian counts). In this experiment, we do not consider the direction of changes when comparing our anomalies with the ground truth as we did in the experiments in Section 3.4. The main reason is due to the involvement of multiple locations and the changes in pedestrians are not homogeneous across all the locations as observed in Figure

Table 3.16 Ground truth events for anomaly detection during morning period from 7am to 9am.

Events	Date	Ground truth hours	Expected change
Queen's Birthday	Mon, 9 June 2014	7am-9am	Decreased
Melbourne Cup	Tue, 4 November 2014	7am-9am	Decreased
Holiday period	Mon, 22 December 2014	7am-9am	Decreased
Holiday period	Tue, 23 December 2014	7am-9am	Decreased
Holiday period (day before Christmas)	Tue, 24 December 2014	7am-9am	Decreased
Christmas Day	Thu, 25 December 2014	7am-9am	Decreased
Boxing Day	Fri, 26 December 2014	7am-9am	Decreased
Holiday period	Mon, 29 December 2014	7am-9am	Decreased
Holiday period	Tue, 30 December 2014	7am-9am	Decreased
Holiday period (day before new year)	Wed, 31 December 2014	7am-9am	Decreased

3.8. In addition, we are unable to propose a more detailed set of ground truth events with expected changes in pedestrian traffic for all locations.

Anomalies during morning peak

We have a span of three hours for each of the anomalies during the morning peak as shown in Table 3.16 and use the entire span to rank how well an anomaly is detected by our approach. To this end, if the algorithm detects all three hours of an anomaly, we conclude that the anomaly is fully detected; the result is less reliable if there is less overlapping of the hours being detected and the hours in the ground truth. The anomalies detected during the morning peak period are listed in Table 3.18. It shows that our approach can either fully or partially detect all 10 anomalies that we select. The anomalies that correspond to actual public holidays are fully detected at all 3 hours. This is sensible since there is arguably much less pedestrian traffic as most of the businesses are closed in the CBD during this time. On Boxing Day, the hour at 9am is not detected as an anomaly, indicating that the pedestrian activities do not drop at this hour. Although Boxing Day is a public holiday, it is the biggest shopping day of the year with various sales. One explanation for this phenomenon is that many people still go to CBD in the morning (but not as early as normal working days) for shopping. For other days during the year end holiday period, our approach did not find any anomalies at 7am. When looking at the categorized pedestrian distributions (`LOW`, `MED`, `HIGH`) at 7am of these days, they indeed are not different from the distributions at 7am of other days of the week at all locations except for Melbourne Central.

Anomalies during evening

Table 3.17 Ground truth events for anomaly detection during evening period from 10pm to midnight.

Events	Date	Ground truth hours	Expected change
Melbourne festival	Fri, 10 Oct 2014	10pm-midnight	Increased
Halloween event	Fri, 31 Oct 2014	10pm-midnight	Increased
Cricket Match	Fri, 7 Nov 2014	10p-11pm	Increased
Football match	Wed, 12 Nov 2014	10p-11pm	Increased
Equestrian event	Thu, 20 Nov 2014	10p-11pm	Increased
Cricket Match	Fri, 21 Nov 2014	10p-11pm	Increased
Football match	Thu, 27 Nov 2014	10p-11pm	Increased
Food festival	Fri, 5 Dec 2014	10p-11pm	Increased
Christmas' Eve	Wed, 24 Dec 2014	10pm-midnight	Increased
New Year's Eve	Wed, 31 Dec 2014	10pm-midnight	Increased

Table 3.18 Anomalous events detected during morning peak periods.

Events	Date	Hours that are detected	Count of hours that are detected (out of 3)
Queen's Birthday	9 Jun 2014	7am, 8am, 9am	3
Melbourne Cup	4 Nov 2014	7am, 8am, 9am	3
Christmas Day	25 Dec 2014	7am, 8am, 9am	3
New Year's Eve	31 Dec 2014	7am, 8am, 9am	3
Boxing Day	26 Dec 2014	7am, 8am	2
Holiday period	29 Dec 2014	8am, 9am	2
Holiday period	30 Dec 2014	8am, 9am	2
Holiday period	22 Dec 2014	8am	1
Holiday period	23 Dec 2014	9am	1
Christmas Eve	24 Dec 2014	9am	1

During the midnight period, 91 anomalies spread over 60 days were detected. Table 3.19 summarizes the results when comparing the anomalies with our ground truth. All events are detected, with the New Year's Eve, Christmas Eve, and Halloween being fully detected, and the rest being partially detected. The ground truth we propose in Table 3.17 only covers 24 anomalies (the total number of hours of all events) and our approach is able to detect 18 out of these. In addition, the other 73 anomalies that are not justified by the ground truth are not yet confirmed to be all false positives, due to the limited size of our ground truth. We group all anomalies into three categories as follows.

- 18 anomalies can be matched with an event in our ground truth.
- 56 anomalies fall on Fridays, that do not match with any event.

Table 3.19 Anomalous events verified with ground truth during evening periods.

Events	Date	Hours that are detected	Counts of hours that are detected
New Year's Eve	31 Dec 2014	10pm, 11pm, midnight	3
Christmas' Eve	31 Dec 2014	10pm, 11pm, midnight	3
Halloween event	31 Oct 2014	10pm,11pm, midnight	3
Melbourne festival	10 Oct 2014	10pm, 11pm	2
Food festival	5 Dec 2014	10pm	1
Cricket Match	7 Nov 2014	10pm	1
Football match	12 Nov 2014	10pm	1
Equestrian event	20 Nov 2014	10pm	1
Cricket Match	21 Nov 2014	10pm	1
Football match	27 Nov 2014	10pm	1

- 17 anomalies that do not fall into the two categories above.

There are 17 anomalies that we cannot match to any major events being publicly advertised. They are spread over all 6 months from Jun to Dec, occur at all weekdays with almost equal frequencies. However, they are not necessarily false positives. The unusual distribution of pedestrians could have been caused by the weather, transportation disruptions, or any unofficial events, for which it is challenging to find references.

It is also observed that most of these anomalies happen on Friday nights. It is natural that more pedestrians are expected to show up and stay in the city area on Friday nights, causing anomalous distributions that are not normally observed on other weekdays. To this end, we subsequently use Fridays as labels to quantify the effectiveness of our algorithm.

There are 30 Fridays from 1^{st} Jun to 31^{st} Dec 2014. We first establish a new set of ground truth events of size 90 that contains all three hours (10pm, 11pm, midnight) of each Friday. For the purpose of simplicity, we do not distinguish between anomalies on a normal Friday and anomalies that correspond to an event on Friday: as long as we have an anomaly on Friday, it is considered a true positive. The full table of the ground truth, together with predicted anomalies are shown in Table A.1 in the Appendix. Since we only have positive labels of anomalies, we do not focus on precision (precision is always 1.0 in this case), F1 score, or show the confusion matrix. Our approach achieves the recall score of **0.71**, indicating that it can effectively detect anomalies from the set of Friday nights as ground truth.

One of the limitations of this validation is that it has a very limited set of anomalies that correspond to arguably simple patterns of unusual behaviour, which is an overall increase of pedestrians across all stations. If a more extensive set of ground truth events is available that

describes a larger variety of events and the direction of changes in individual locations, the algorithm can be evaluated more thoroughly on its ability to detect more complex anomalies. In addition, another limitation of our experiment is its narrow windows of three hours, which is used to overcome the challenges of low granularity coming from the categorization of raw count values into three buckets of distributions.

3.4.5 Time Complexity

In this section we discuss the time complexity of the algorithm when it is applied on N observations of pedestrian counts collected by sensors at l locations.

The whole process of modeling the normal behavior and finding anomalies can be broken down into three stages:

- The data normalization traverses all the data points twice, first to find the maximum values of each window then to assign each pedestrian count value into buckets of `LOW`, `MEDIUM` and `HIGH`. This process is performed separately for each location. The time complexity of the data normalization step is $O(N \times l)$.
- The mining of frequent itemsets is repeated multiple times with gradually decreasing minimal support, until all three constraints in Section 3.4.2 are satisfied. This step takes the majority of running time and will be discussed in greater detail below.
- The testing of the coverage of the frequent itemsets on the input data in each iteration traverses all data points. Each traversal checks the overlap of items of that data point and items in the frequent patterns. This is repeated for all the frequent itemsets found. The time complexity is therefore $O(N \times f \times \bar{l})$, where f is the number of frequent itemsets found ($f \ll N$), and $\bar{l}$ is the average number of items in the frequent itemsets.

Regarding the running time of the Apriori algorithm, according to Pang-Ning et al. [210], multiple factors can affect its complexity, including the minimal support threshold, dimensionality of the transactional dataset (number of locations), and number of observations. Its time complexity is mainly affected by the candidate generation step, i.e., k-itemsets are generated from (k-1)-itemsets. The time complexity of frequent pattern mining in its brute-force approach [210] is $O(2^l)$. This is significantly reduced with candidate pruning to $O(\Sigma_k(k \times F_{k-1} \times F_1))$, where F_1 and F_{k-1} are the number of frequent items, and the number of frequent (k-1)-itemsets respectively. These factors depend on the input and are not trivial to derive, however they can still be in the worst case exponential with respect to the number of transaction items. In addition to this, our algorithm executes the Apriori algorithm several times until all the constraints are satisfied. The number of iterations as well as the values

of the trial supports can only be known in retrospect. It is hence challenging to analytically formulate the time complexity of the algorithm.

We therefore empirically analyze the effect of the number of observations N and the dimensionality l on the time complexity. We test the algorithm on four subsets of data: 3 months, 6 months, 9 months and 12 months of data, and observe that there is little variation in the observed execution times as the number of observations increases. This is inline with the fact that the time complexity is dominated by the candidate generation, which in turn is mainly affected by the number of items (locations in our case).We then test the algorithm on data with varying numbers of locations (dimensions), ranging from 4 to 28 dimensions. For each dimension, the algorithm is applied on all four subsets of the data mentioned above. The average execution time for each number of dimensions is presented in Table 3.20 and plotted in Figure 3.9.

Figure 3.9 shows that the execution time grows exponentially with the number of dimensions. In order to better understand the nature of the time complexity, we then attempt to formulate the function of running time w.r.t the number of locations l. Our running time function is also plotted in Figure 3.9 and has the form of:

$$f(l) = a \times 2^{b \times l} + c$$

where a indicates the number of iterations of performing the Apriori algorithm to find frequent itemsets, b provides some insights on how much the pruning of the Apriori algorithm helps reduce the time complexity, and c is a constant that expresses any overhead of the algorithm execution.

By fitting the function to the empirical running time, we find that:

- **a=13.0**: Our algorithm executes the Apriori algorithm 13 times, with minimal support gradually reduced from 100% to 40%. This indicates that in this time complexity analysis, the algorithm reaches our minimal support threshold of 40% before it can find any frequent itemsets that satisfy all 3 constraints in Section 3.4. From Figure 3.9, this might not apply to the cases where the number of dimensions is less than 16: the actual running times are smaller, and fewer iterations of Apriori might have been executed for these cases.

- **b=0.498**: Without pruning, the time complexity of each execution of brute force frequent pattern mining is $O(2^l)$, i.e., $b = 1$. The value of b indicates that Apriori reduces the exponential numbers of candidates. However, the time complexity still remains exponential to the numbers of locations (as also observed from Figure 3.9), which makes scaling this algorithm to higher numbers of locations a challenge.

- **c=8970**: This can be accounted for by any overhead of our implementation of the algorithm, including loading, logging, etc.

Dimensionality (number of locations)	Execution time (ms)
4	390
8	630
10	651
12	1740
14	2954
16	5960
18	24987
20	25044
22	29675
24	104241
26	104227
28	190620

Table 3.20 Execution time of the algorithm with different numbers of dimensions.

3.4.6 Summary for the Frequent Pattern-Based Approach

In this section, we present our attempt to address the problem of detecting anomalies in data having more than two dimensions. We use frequent pattern mining as the underlying technique to form a model of the pedestrian count distribution across ten locations. The model was then used to extract anomalous events from the dataset, which were subsequently validated against known events.

Our proposed algorithm manages to tackle the problem of detecting anomalies in data with more than two dimensions, which HyCARCE and ESM fail to do as we mentioned in Section 3.3. The anomalies are validated against our ground truth of known public holidays and events, the results show that the algorithm can either fully or partially detect the anomalies during morning peaks and evening. We subsequently use Friday evenings as another source of ground truth to quantify the effectiveness of the algorithm and show that it can achieve a high recall score. The high level verification and concrete verification provide evidence that the algorithms can detect anomalies during morning peaks and evening time effectively.

However, there are limitations with our proposed approach. First, the categorization of raw counts of pedestrians into just three categories reduces the granularity of the input, and subsequently the result. The immediate consequence is reflected in the small sizes of windows of three hours we use in this experiment, which limits our analysis to a certain period

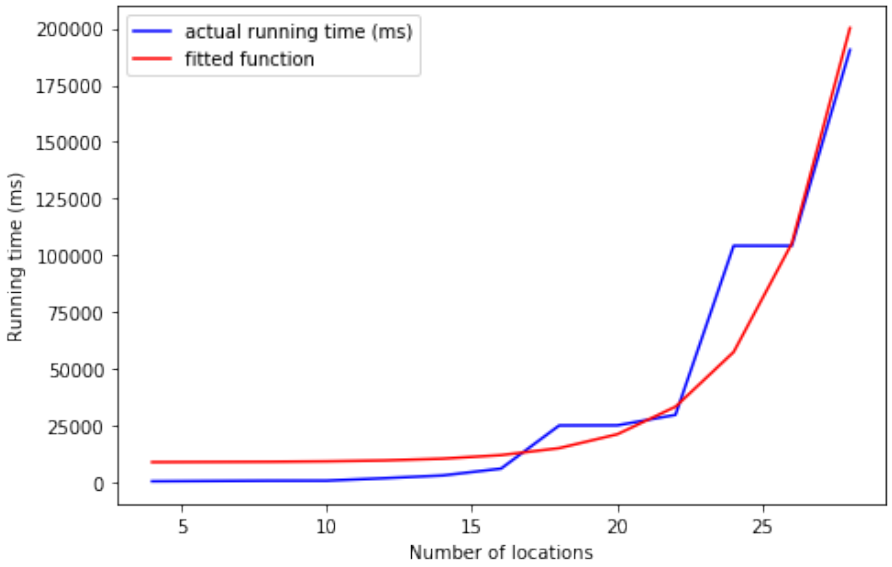


Fig. 3.9 Execution time of the algorithm with different dimensionalities.

of the day only. Performing the same experiment with larger window sizes did not yield any insightful results, and it remains a challenge with this approach. By learning the normal behavior in separate windows, the patterns found cannot be directly compared (because they are normalized in their own windows); therefore, their similarities and differences at different hours of the day, or how they develop over time are not revealed.

Another limitation of this approach lies in its high time complexity. Invoking multiple iterations of Apriori is expensive and affects its scalability to higher numbers of locations.

3.5 Summary

This chapter presents two approaches to model the pedestrian distribution of the City of Melbourne in both low and medium dimensional data space. We also discuss and illustrate the challenges that emerge when analyzing higher dimensional data. Specifically, the main challenge is the local feature relevance that makes traditional similarity measures ineffective, for which we resort to categorization and frequent pattern mining as a workaround to construct the model.

Intuitively, this approach of categorization classifies the original count values into much fewer buckets to simplify the comparison. The Apriori algorithm finds the items (locations) whose attributes are frequently similar to form the frequent patterns. In the context of subspace clustering, this is equivalent to finding the dimensions (locations) that are strongly

correlated and should belong to the same subspace. To this end, the frequent-pattern-based approach can be considered a solution that addresses the problem of *local feature relevance* by compromising the granularity of data (by categorization), while still suffering from high time complexity and scalability to higher dimensional data. This raises the need for using subspace clustering to find the patterns of data without categorizing or discretizing.

In the following chapters we present our proposed methods to measure similarity effectively in high dimensional spaces and subsequently to cluster high dimensional data. We then demonstrate empirically how clustering high dimensional data without discretizing or categorizing can construct a model with greater detail and provide more valuable findings.

Chapter 4

An Effective Measure of Similarity in High Dimensional Space

Measuring the similarities between points is a critical task that has a direct impact on clustering results. This task becomes increasingly challenging as the number of dimensions of the data space increases. In high dimensional space, not all dimensions are relevant when measuring the similarity between data points. As we illustrate in this chapter, just a small proportion of irrelevant dimensions can significantly alter the distances between data points and obfuscate their similarities. We propose that the similarity between points in high dimensional space is reflected in their similarities in the component low dimensional subspaces. We provide the proof of concept for euclidean-based, density-based, and grid-based similarity. Subsequently, we propose our algorithm Local Similarity to measure similarities in high dimensional space. Our evaluation shows that the proposed algorithm can determine the relevant subspace for each pair of data points, and can find their similarity better than traditional measures. We analyze the time complexity and subsequently provide a derived version of the algorithm with sampling approach that is significantly less computationally expensive while still providing comparable results.

The similarity measure discussed in this chapter has been submitted in the following paper:

M. T. Doan, J. Qi, S. Rajasegarar, and C. Leckie. Scalable Bottom-up Subspace Clustering using FP-Trees for High Dimensional Data. Proceedings of the 6^{th} IEEE International Conference on Big Data (IEEE BigData), 2018

4.1 Introduction

As clustering is a task of grouping similar points into the same group, measuring the similarity between data points is a critical step that directly affects the effectiveness of clustering algorithms [110]. The most common similarity measurements are Euclidean distance and Manhattan distance, which are based on the geometrical distance between data points. However, multiple studies have shown that these traditional measurements become ineffective in high dimensional space [12, 47, 85, 192]. In fact, measuring similarity between data points raises multiple challenges as the number dimensions increases. We summarize the challenges as follows:

1. The most significant challenge is the property of *local feature relevance*, which states that points are usually correlated only in different subsets of dimensions. Therefore, not all dimensions are relevant when computing the similarity between points, otherwise the similarity can be hindered by irrelevant dimensions. For example, adding an irrelevant dimension in which the measurements are significantly different can spread two similar points apart and obfuscate the similarity in more relevant dimensions.

2. The second challenge is caused by the curse of dimensionality. Using traditional similarity measurements, the points are observed to be spread further apart, to an extent where they can be considered equidistant. These two challenges are discussed in greater detail in Chapter 2.

3. The similarity between two points is decided by multiple features (dimensions) measuring different attributes. Aggregating the similarity of each attribute into a single concise and holistic similarity measure remains a research challenge [200]. Multiple studies have pointed out that the similarity between two points does not just depend on the two points by themselves but also the applications and the distributions of data [152, 83]. An effective application needs to take into account these factors when computing similarities between points in high dimensional space.

Different techniques have been proposed to tackle these challenges, including PCA [117], MDS [35], manifold learning [173, 245], and feature selection techniques [43]. The most common approach is to use these techniques to reduce the number of dimensions before using traditional similarity measurements to compute the similarity between points in the latent subspaces [108, 209]. This approach addresses the curse of dimensionality. However, it refines the dimensions in a global fashion and fails to learn the local feature relevance between data points in different subspaces [128]. Therefore, irrelevant dimensions can still potentially spread two similar points far apart as we show in an example later in this chapter.

We first prove that the proximity between two points in high dimensional space is constrained and can be inferred from the proximity of these points in its lower dimensional subspace components. This approach of aggregation of low dimensional similarities enables our measurement of similarity in high dimensional space that can overcome the challenges of local feature relevance. We discuss the advantages of our algorithm and show that it can find relevant subspaces, and discard irrelevant dimensions, in order to compute the similarities between points. Our evaluation in Section 4.5 shows that it outperforms existing traditional similarity measurements.

4.2 Related Work

Weighted Euclidean distance [184, 237] and weighted cosine distance [134] can find meaningful similarities between points if the dimensions in which different groups of points are correlated are known in advance. To this end, high weights can be assigned to these dimensions and low (or zero) weights can be assigned to irrelevant dimensions. However, this knowledge is generally not available in advance in practice. In fact, in the context of subspace clustering, finding an appropriate set of weights for all dimensions is the main challenge that top-down clustering algorithms address as discussed in Chapter 2. Therefore, without this knowledge, weighted distances cannot be used as stand-alone similarity measurements in high dimensional space.

Among traditional metrics, cosine similarity is a simple measure that possesses multiple advantages in measuring similarities in high dimensional space [178]. Each point is considered as a high dimensional vector and the similarity between two points is determined by the angle between two corresponding vectors. The cosine similarity between random vectors are distributed around 0, so the significance of correlation (albeit small) is amplified as the number of dimensions increases. This metric is good at capturing the similarity of patterns of feature changes, e.g., the proportional changes in the values of features while ignoring the exact magnitude of changes, which is useful for many applications, such as informatics, and multivariate phenotype analysis [240]. Cosine similarity uses all dimensions and hence irrelevant dimensions can distort the meaningful angle between two points, in contrast to when the similarity is considered only in the relevant subspaces.

A common approach to manage high dimensionality is to first reduce the number of dimensions and then compute the similarity in the lower dimensional spaces. Early dimensionality reduction techniques include PCA, Independent Component Analysis (ICA) [132], and Linear Discriminant Analysis (LDA) [112]. These techniques find a linear projection of data onto a lower dimensional space. In addition, many techniques of manifold learning have

been proposed to handle data with non-linear structures. These techniques include MDS, ISOMAP [212], Local Linear Embedding (LLE) [187], Hessian Eigenmapping [73], and t-SNE [143, 220]. These techniques project data onto a lower dimensional subspace, which partially reduces the effect of the curse of dimensionality. They can be applied on data having different sizes, structures, and distributions in a wide range of applications. However, all the aforementioned techniques apply global changes to the data and result in only one final lower dimensional space, effectively assuming all points to be correlated in only one subspace.

One of the main reasons why the previous methods are ineffective is that they are not able to find local relevance between features in high dimensional space; therefore, attributes in irrelevant dimensions can distort the similarities. In fact, different applications show different algorithmic approaches to assess the similarities between data points. For example, bottom-up clustering algorithms find dense units in each dimension and exploit the downward closure property of density to aggregate these dense units to form dense areas in high dimensional space. Another example is the top-down clustering algorithms that assign lower weights to irrelevant dimensions and higher weights to more relevant ones. The measurements of similarities between data points in these applications are implicit, and tightly coupled to the applications. In this chapter, we propose a more generic method to measure similarities in high dimensional space by first identifying relevant dimensions then subsequently computing a proximity that better reflects the similarity between data points. We discuss and evaluate our similarity measurement in this chapter and demonstrate in the next chapter how our proposed technique can be used to develop a novel subspace clustering algorithm.

4.3 Proposed Method

In this section we present the theoretical foundation to support our similarity measurement and subsequently propose our algorithm *Local Similarity* to compute the similarity between two points in high dimensional space. We first define the concepts and terminology used to develop our proposed method

- **Base subspace** is a low p-dimensional subspace that has all dimensions being relevant to the similarity between two points. Therefore, all dimensions equally contribute to the computation of similarity. The number of dimensions of a base subspace is low to avoid the local feature relevance problem, in order to guarantee the relevance of all component dimensions. We use $p = 2$ in this thesis; we discuss in the next sections that this value balances the complexity and can work with a wide variety of datasets. If more knowledge about the dataset is available, p can be adjusted accordingly. For

example, if it is known in advance that data collected by three nearby sensors can all be taken into account to compute similarity, the base subspace can be 3-dimensional.

- **Base similarity** of two points is their similarity computed using a traditional distance metric (e.g., Euclidean, cosine similarity, etc.) in a base subspace.
- **Locally relevant subspace (LRS)** is a subspace whose dimensionality is higher than that of a base subspace, that has all dimensions being relevant and contributing to the similarity computation between two points. We show in this chapter that we construct a LRS by aggregating relevant *base subspaces* together.

We prove that the aggregation of *base similarities* between two points can preserve and reflect their similarity in the aggregated subspace. This approach enables us to first find relevant base subspaces in which the *high similarity* between two points are exhibited, then subsequently aggregate these base subspaces to form the LRS. By definition, all dimensions in a LRS are relevant and contribute to the similarity computation, hence any traditional distance metrics can be applied without suffering from the local feature relevance challenge. This premise is the basis of our proposed *Local Similarity* algorithm.

Next, we present the underlying theory of forming high dimensional similarity by aggregating similarities in low dimensional base subspaces, and subsequently present our proposed algorithm as shown in Algorithm 1. Specifically, we show the proof of concept for aggregation of euclidean-based similarity, grid-based similarity, as well as similarity based on the density of the data. We focus on *base similarities* in $2D$ subspaces, i.e., $p = 2$. The reason for this choice of parameter is elaborated in greater detail in the next section. We only present the proofs under this case, although the proofs can easily be extended to more generic cases of base similarities in subspaces with other dimensionalities.

4.3.1 Proof of Concept for Euclidean-based Similarity

Lemma 4.3.1 *Given two points x_i, x_j in a k-dimensional subspace $S^{(k)}$, if x_i and x_j are near each other in every $2D$ subspace $S^{(2)'} \subset S^{(k)}$, i.e., $d_{S^{(2)'}}(x_i, x_j) < r$, then the distance between the two points in $S^{(k)}$ is proportional to r and grows in the order of $O(\sqrt{k})$.*

Proof: Without loss of generality, assume that point x_j is the origin, and the threshold of distance between x_i and x_j to be considered close in every $2D$ subspace $S^{(2)'} \subset S^{(k)}$ is r. These assumptions can be easily met by normalizing the coordinates and linearly transforming each dimension.

Since the distance between x_i and x_j (the origin) in every $2D$ subspace is less than r, the point x_i is located in a circle centered at the origin with a radius of r. The coordinates of x_i in the component dimensions of $S^{(k)} = \{d_{i1}, d_{i2}, ..., d_{ik}\}$, denoted by x_{i1}, x_{i2}, ..., x_{ik}, satisfy:

$$\begin{aligned} \forall q,t|\{d_q,d_t\} \subset S^{(k)} : x_{iq}^2 + x_{it}^2 &\leq r^2 \\ \text{Hence, } x_{i1}^2 + x_{i2}^2 &\leq r^2 \\ x_{i1}^2 + x_{i3}^2 &\leq r^2 \\ &... \\ x_{ik-1}^2 + x_{ik}^2 &\leq r^2 \end{aligned}$$

There are $\frac{k(k-1)}{2}$ combinations of $2D$ subspaces and hence $\frac{k(k-1)}{2}$ inequalities. Each term x_{iq} appears for $k-1$ times. Summing both sides of the above inequalities yields:

$$\begin{aligned} (k-1)(x_{i1}^2 + x_{i2}^2 + ... + x_{ik}^2) &\leq \frac{k(k-1)}{2} r^2 \\ \text{Thus, } \sqrt{x_{i1}^2 + x_{i2}^2 + ... + x_{ik}^2} &\leq r\sqrt{\frac{k}{2}} \end{aligned} \tag{4.1}$$

Therefore, a hypersphere centered at the origin with a radius of $r^{(k)} = r\sqrt{\frac{k}{2}}$ is sufficient to bound the point x_i inside.

We have shown that two points are near each other in a k-dimensional space $S^{(k)}$ if they are near in every $2D$ subspace of $S^{(k)}$. We relax this assumption in Lemma 4.3.2 and show that the sublinear growth of order $O(\sqrt{k})$ of the radius still holds.

Lemma 4.3.2 *Given two points x_i, x_j in a k-dimensional subspace $S^{(k)}$, if x_i and x_j are near each other (i.e., $d(x_i,x_j) < r$) in a certain set of $2D$ subspaces $\{S^{(2)'} | S^{(2)'} \subset S^{(k)}\}$ such that each component dimension is involved at least α ($\alpha \geq 1$) times, then the distance between x_i and x_j in $S^{(k)}$ is proportional to r and grows in the order of $O(\sqrt{k})$.*

Example: Consider two points x_i, x_j in a 4-dimensional subspace $S^{(4)} = \{d_1, d_2, d_3, d_4\}$. If these two points are near (within distance r) in the subspaces $\{d_1, d_2\}$, $\{d_2, d_3\}$, $\{d_3, d_4\}$, $\{d_1, d_4\}$, each component dimension appears twice, i.e., $\alpha = 2$. On the other hand, if x_i and x_j are near in every $2D$ subspace $\{d_1, d_2\}$, $\{d_1, d_3\}$, $\{d_1, d_4\}$, $\{d_2, d_3\}$, $\{d_2, d_4\}$, $\{d_3, d_4\}$, it is a special case which is described by Lemma 4.3.1, in which $\alpha = k-1$, where k is the number of dimensions.

Proof: Without loss of generality, assume x_j is the origin. Since the distance between x_i and x_j is less than r in some $2D$ subspaces, we have:

$$\begin{aligned} \exists q,t|\{q,t\} \subset S^{(k)}, x_{iq}^2 + x_{it}^2 \leq r^2 \\ \text{Hence, } x_{i1}^2 + x_{i2}^2 \leq r^2 \\ ... \\ x_{ik-1}^2 + x_{ik}^2 \leq r^2 \end{aligned} \tag{4.2}$$

Here, each term x_{iu} appears at least α times. There are also some x_{iv} that appear more than α times. We denote *sum_extra* as the sum of all the residues after counting each term α times. Summing the inequalities yields:

$$\begin{aligned} \alpha(x_{i1}^2 + x_{i2}^2 + ... + x_{ik}^2) + sum\_extra \leq mr^2 \\ \text{Therefore, } x_{i1}^2 + x_{i2}^2 + ... + x_{ik}^2 \leq \frac{m}{\alpha} r^2 \end{aligned} \tag{4.3}$$

Here, m is the number of inequalities. As r^2 and α are known, finding the upper bound of the radius of the hypersphere in the k-dimensional subspace to bound the point x_i is equivalent to finding the maximum value of m that guarantees that (1) each term x_i appears at least α times and, (2) not every term appears more than α times. This can be formulated as a problem in graph theory. All inequalities in Equation 4.2 are represented by a graph $G = (V,E)$. Each term x_{iq} is represented by a vertex $v_q \in V$. Each inequality in the form $x_{iq}^2 + x_{ir}^2 \leq r^2$ is represented by an edge e_{qr} between vertices v_q and v_r. Thus, $m = |E|$. Finding the maximum value of m becomes the problem of finding a graph with the maximum number of edges such that at least a vertex has a degree of α:

$$max(|E|) \text{ s.t. } \delta(G) = \alpha, \text{ where } \delta(G) \text{ is the minimum degree [162] of } G. \tag{4.4}$$

The number of edges can be maximized by connecting vertex e_1 to $k-1$ other vertices, e_2 to $k-2$ other vertices (except e_1), ..., e_α to $k-\alpha$ other vertices (except the first $\alpha-1$ vertices which are already connected to e_α). This ensures the degree of at least one vertex is α, hence $\delta(G) = \alpha$.

The maximum value of m is then expressed as:

$$m_{max} = (k-1) + (k-2) + ... + (k-\alpha) = \sum_{i=1}^{\alpha} k - \alpha = \alpha k - \frac{\alpha(\alpha+1)}{2} \tag{4.5}$$

Replacing m_{max} in Equation 4.3, we can derive:

$$r^{(k)} \leq r\sqrt{k - \frac{\alpha+1}{2}} \tag{4.6}$$

Equation 4.6 shows that, given the proximity of x_i and x_j in some $2D$ subspaces, the distance between the two points is proportional to the distance r in the component $2D$ subspaces and this distance grows in a sublinear order with respect to the increase of the number of dimensions. It can also be observed that in the ideal case where $\{x_i\}$ is clustered in every $2D$ subspace, $\alpha = k-1$, Inequality (4.6) becomes Inequality (4.1). Equation (4.6) expresses the constraints of the spatial distance between the points in the higher dimensional subspace in terms of its proximity in the component $2D$ subspaces. It validates our algorithm by showing that higher dimensional similarities can be constructed by aggregating $2D$ base similarities. *Intuitively, if two points are similar in a certain set of base subspaces, they are also similar in the higher dimensional data space that is constituted of these lower dimensional subspace dimensions. We consider this higher dimensional space a locally relevant subspace in which coordinates of all component dimensions contribute to the similarity between points.*

4.3.2 Proof of Concept for Density-based Similarity

We follow a probabilistic approach together with the downward closure property of density to guarantee the validity of aggregating base similarities to form overall similarity in higher dimensional subspaces. This is formulated in Lemma 4.3.3.

Lemma 4.3.3 *Given two points x_i and x_j in k-dimensional data space $S^{(k)}$, the similarity between x_i and x_j in $S^{(k)}$ is proportional to the cardinality $|\{S'\}|$ (with each S' being a subspace of $S^{(k)}$), such that x_i and x_j are located in the same dense area in S'.*

Proof: Let $C_{S^{(k)}}$ denote the event that two points x_i and x_j belong to the same dense area in the subspace $S^{(k)}$. Assume that we also know that these two points belong to the same dense areas in a set of p subspaces $\mathscr{S}_1 = \{S_1, ..., S_m, ..., S_p\}$ ($S_m \subset S^{(k)}$). Given this knowledge, the

probability of $C_{S^{(k)}}$ (x_i and x_j belong to the same dense area in $S^{(k)}$) is formulated as:

$$P_1 = P(C_{S^{(k)}} | C_{S_1}, ..., C_{S_p}) = \frac{P(C_{S^{(k)}}, C_{S_1}, ..., C_{S_p})}{P(C_{S_1}, ..., C_{S_p})} \tag{4.7}$$

We show that the probability P_1 increases as new evidence of proximity between x_i and x_j is found in other subspaces of S_k. Specifically, let these two points also be near in a certain new subspace $S^* \subset S_k$ ($S^* \not\subset \mathscr{S}_1$, i.e., S^* is indeed a newly discovered subspace in which x_i and x_j belong to the same dense area). The probability of them belonging to the same dense area in $S^{(k)}$ now becomes:

$$P_2 = P(C_{S^{(k)}} | C_{S_1}, ..., C_{S_p}, C_{S^*}) = \frac{P(C_{S^{(k)}}, C_{S_1}, ..., C_{S_p}, C_{S^*})}{P(C_{S_1}, ..., C_{S_p}, C_{S^*})} \tag{4.8}$$

By applying the chain rule, we proceed to show that $P_2 > P_1$.

$$\frac{P_2}{P_1} = \frac{P(C^{(k)}, C_{S_1}, ..., C_{S_p}, C_{S^*})}{P(C_{S_1}, ..., C_{S_p}, C_{S^*})} \times \frac{P(C_{S_1}, ..., C_{S_p})}{P(C_{S^{(k)}}, C_{S_1}, ..., C_{S_p})} \tag{4.9}$$

According to the downward closure property of density, if x_i and x_j are near in subspace S_k, they are also near in all subspaces of $S^{(k)}$, including S^*. Hence $P(C_{S^*}|C_{S^{(k)}}) = 1$, or $P(C_{S^*}, C_{S^{(k)}}) = P(C_{S^{(k)}})$. Therefore, $P(C_{S^{(k)}}, C_{S_1}, ..., C_{S_p}, C_{S^*}) = P(C_{S^{(k)}}, C_{S_1}, ..., C_{S_p})$. Equation 4.9 can then be rewritten as:

$$\frac{P_2}{P_1} = \frac{P(C_{S_1}, ..., C_{S_p})}{P(C_{S_1}, ..., C_{S_p}, C_{S^*})} = \frac{\sum_{C_{S^*}} P(C_{S_1}, ..., C_{S_p}, C_{S^*})}{P(C_{S_1}, ..., C_{S_p}, C_{S^*})} \geq 1 \tag{4.10}$$

By marginalising the numerator over C_{S^*}, we can deduce that $\frac{P2}{P1} \geq 1$. We therefore show that additional evidence of x_i and x_j belonging to the same dense area in another lower dimensional subspace $S^* \subset S^{(k)}$ increases the probability that these two points belong to the same cluster in S_k. Thus Lemma 4.3.3 is proved.

The intuition of Lemma 4.3.3 is that the formation of clusters in lower dimensional subspaces can be used as evidence to reinforce and increase the posterior probability of the formation of a dense area for the same set of points in the higher dimensional super subspaces. Therefore, *we say that if a set of points are dense in a certain set of base subspaces, they are also dense in the higher data space that is constituted of the component dimensions of these base subspaces. We consider this higher dimensional space a locally relevant subspace*

in which traditional distance metrics can be used to measure the similarity between points without suffering from the local feature relevant challenge.

4.3.3 Proof of Concept for Grid-based Similarity

In grid-based clustering algorithms [17, 125, 197], the range of values in each dimension is equally divided into g bins, resulting in a total of g^d hyper-cells in a d-dimensional subspace. To this end, a group of points are similar to each other if they are closely distributed, i.e., they only occupy a small number of cells in the entire space. The constraints of proximity of a set of points X_j can be defined by the criterion as follows:

$$occupancy\_ratio = \frac{cell^{(d)}(X_j)}{g^d} \leq T \tag{4.11}$$

$$\text{where } cell^{(d)}(X_j) = \prod_{i=1}^{d} \frac{max(X_{ji}) - min(X_{ji})}{g} \tag{4.12}$$

Here, X_{ji} contains the projections of points X_j onto the dimension i, $cell^{(d)}(X_j)$ is the *upper bound* number of hyper-cells that the set of points X_j occupy in this d-dimensional subspace, T is a threshold that indicates the proportion of hyper-cells this set of points occupy. Note that we use the upper bound number of cells in Equation 4.11 instead of the actual number of cells, in order to ensure cell adjacency. Intuitively, $cell^{(d)}(X_j)$ can be regarded as the *hyper-cube* region in high dimensional space that is bounded by the upper bounds and lower bounds of the projections of the points onto each dimension. The actual number of cells occupied by the points can be lower. T is used to control the density of points that are considered similar. Figure 4.1 illustrates an example of 300 points in a 3-dimensional data space limited by the ranges $[-1,1]^3$. The range in each dimension is divided into 10 equal bins, resulting in $10^3 = 1,000$ cells. The points occupy only 27 out of 1,000 cells, i.e., $occupancy\_ratio = 0.027$. If T is set to 0.05, these points are considered similar.

We propose Lemma 4.3.4 to show that the similarity between points in high dimensional space can be constructed by the similarities between these points in the component lower dimensional subspaces.

Lemma 4.3.4 *If a set of points X_j are similar in some sets of distinct 2-dimensional subspaces $\{S_1, S_2, ..., S_i\}$ (i.e., $\forall u, v \in [1,i], S_u \neq S_v$), then these points are similar in the higher dimensional subspace $\mathscr{S} = \bigcup_{m=1}^{i} S_m$*

Proof: Let X_j be a set of points in a d-dimensional subspace. The range in each dimension is equally divided into g bins. Let X_j be similar to each other when projected on some $2D$

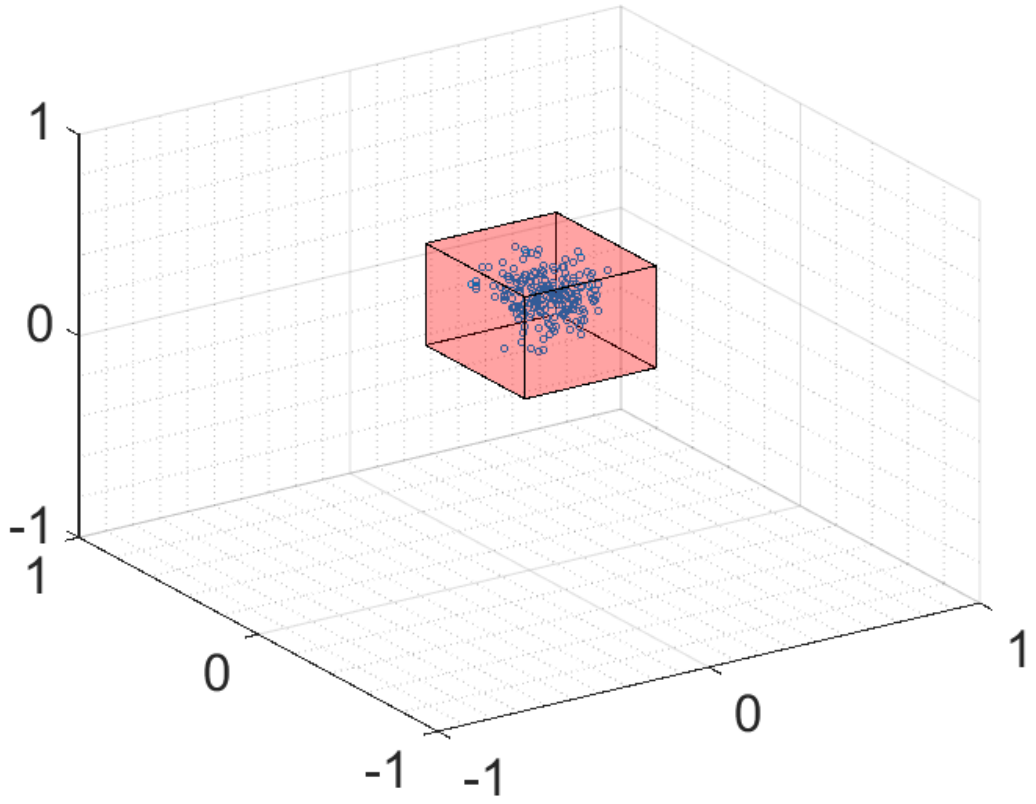


Fig. 4.1 A set of points occupy a small portion of the full data space.

subspaces $\{S_1, S_2, ..., S_i\}$, i.e., they satisfy the condition:

$$occupancy\_ratio = \frac{cell^{(2)}(S_m)}{g^2} \leq T, m \in [1, i] \tag{4.13}$$

We subsequently prove Lemma 4.3.4 by induction. For the base case, we show that if X_j are similar to each other in two $2D$ subspaces S_1 and S_2 then they are also similar in subspace $S_1 \cup S_2$. Specifically, the number of cells occupied by X_j in $S_1 \cup S_2$ is given by $cell(S_1 \cup S_2) \leq cell(S_1) \times cell(S_2)$. On the other hand, since S_1 and S_2 are distinct, the subspace $S_1 \cup S_2$ can be a 3-dimensional subspace (if these two subspaces share one dimension in common), or a 4-dimensional subspace (if S_1 and S_2 are disjoint). Therefore, the total number of cells in the resulting subspace satisfies $\mathscr{G} \geq g^3$. The *occupancy_ratio* is

given by:

$$occupancy\_ratio = \frac{cell(S_1 \cup S_2)}{\mathscr{G}} \leq \frac{cell(S_1) \times cell(S_2)}{g^3} \tag{4.14}$$

$$occupancy\_ratio \leq \frac{cell(S_1)}{g^2} \times \frac{cell(S_2)}{g} \tag{4.15}$$

Since $\frac{cell(S_1)}{g^2} \leq T$ and $\frac{cell(S_2)}{g} \leq 1$, Equation 4.15 becomes $occupancy\_ratio \leq T$. Thus, the points are similar in the resulting subspace when combining two 2-dimensional subspaces. The base case is proven.

Next, for the inductive step, we prove that if points X_j are similar in a d-dimensional subspace $\mathscr{S}_k = S_1 \cup S_2 ... \cup S_k$ and a 2-dimensional S_{k+1} ($S_{k+1} \notin \mathscr{S}_k$), then X_j are also similar in the subspace $\mathscr{S}_{k+1} = \mathscr{S}_k \cup S_{k+1}$. Since X_j are similar in $\mathscr{S}_k$, we have:

$$occupancy\_ratio = \frac{cell^{(d)}(X_j)}{g^d} \leq T \tag{4.16}$$

The problem can be divided into three cases when combining $\mathscr{S}_k$ with S_{k+1}.

Case 1: $\mathscr{S}_{k+1}$=$\mathscr{S}_k$. This happens when each dimension of S_{k+1} already exists in the high dimensional subspace $\mathscr{S}_k$. There is no change in the number of cells the points occupy, or the number of cells of the data space. Equation 4.16 still holds and X_j are similar to each other.

Case 2: $\mathscr{S}_{k+1}$ is expanded by one dimension. The *occupancy_ratio* is given by:

$$occupancy\_ratio = \frac{cell^{(d+1)}(X_j)}{g^{d+1}} \leq \frac{cell^{(d)}(X_j) \times b_{k+1}}{g^d \times g} \tag{4.17}$$

Here, b_{k+1} is the number of bins the data span in the new dimension. As $b_{k+1} \leq g$, Equation 4.17 becomes:

$$occupancy\_ratio \leq \frac{cell^{(d)}(X_j)}{g^d} \leq T \tag{4.18}$$

Thus, the constraint on the occupancy ratio still holds and the points are similar in the resulting subspace.

Algorithm 2: Find locally relevant subspaces

Input : $X \in \mathbb{R}^{n \times d}$
p: dimensionality of base subspace
sim_threshold

Output : $LRS_{n \times n}$: a matrix of locally relevant subspaces of each pair of points.

```
// generate all p-dimensional base subspace
```
1 $all\_base\_subspaces = combination(d, p)$
```
// assess point similarity in each base subspace
```
2 **foreach** *base_subspace in all_base_subspaces* **do**
3 $projected\_X$ = project(X,*base_subspace*)
4 sim_X = point_to_point_distance($projected\_X$)
5 $close\_points$ = filter(sim_X,*sim_threshold*)
6 $LRS_{close\_points} = LRS_{close\_points} \cup base\_subspace$
7 **end**
8 $LRS = remove\_duplicated\_dim(LRS)$
9 **return** *LRS*

Case 3: $\mathscr{S}_{k+1}$ is expanded by two dimensions, i.e., the two dimensions of S_{k+1} have not yet appeared in $\mathscr{S}_k$. The *occupancy_ratio* is given by:

$$occupancy\_ratio = \frac{cell^{(d+2)}(X_j)}{g^{d+2}} \leq \frac{cell^{(d)}(X_j) \times cell^{(2)}(X_j)}{g^d \times g^2} \tag{4.19}$$

$$occupancy\_ratio \leq \frac{cell^{(d)}(X_j)}{g^d} \times \frac{cell^{(2)}(X_j)}{g^2} \leq T^2 \leq T \tag{4.20}$$

Equation 4.20 indicates that the points X_j are similar in the resulting subspace $\mathscr{S}_{k+1}$.

We have proven the inductive step that if the points X_j are similar in the subspace $\bigcup_{m=1}^{k} S_m$ then they are also similar in the subspace $\bigcup_{m=1}^{k+1} S_m$, given that the points are similar in the 2-dimensional subspace S_{k+1}. Lemma 4.3.4 is therefore proven.

4.3.4 Algorithm for Local Similarity

Based on the proof of the concepts presented above, we propose our algorithm Local Similarity to effectively measure the similarity between points in high dimensional subspaces. The idea of our proposed approach is to first determine the *locally relevant subspace* and then apply a traditional distance metric in this subspace to find the *similarity* between two points only in the relevant dimensions. These two processes correspond to Algorithm 2 and Algorithm 3 respectively.

Algorithm 3: Local similarity

Input : $x_1 \in \mathbb{R}^d$
$x_2 \in \mathbb{R}^d$
LRS: locally relevant subspaces of points
underlying_distance_metric

Output : $similarity \in \mathbb{R}$: a scalar value of similarity.

```
// project the coordinates onto locally relevant subspace
```

1 lrs = lookup(LRS, (x_1,x_2))

```
// if no locally relevant subspace exists, use the full
   dimensional space
```

2 **if** *lrs is empty* **then**
3 $projected\_x1 = x_1$
4 $projected\_x2 = x_2$
5 **else**
6 $projected\_x1$ = project(x_1, lrs)
7 $projected\_x2$ = project(x_2, lrs)
8 **end**

```
// compute the distance using the underlying metric in the lrs
   only
```

9 $distance = underlying\_distance\_metric(projected\_x1, projected\_x2)$
10 **return** $(distance, lrs)$

In Algorithm 2, all possible p-dimensional base subspaces are generated from the original high d-dimensional space ($\binom{d}{p}$ subspaces in total). In a given base subspace, if the similarity between two points is below a given threshold (line 5), they are considered similar and the dimensions of the current base subspace are recorded to construct the locally relevant subspaces (line 6). As such, the $filter()$ function on line 5 performs a comparison of each point-to-point similarity in the matrix *sim_X* with the threshold, and returns the pairs of points whose similarity is greater than the threshold. Subsequently, these pairs *close_points* are considered similar in the current *base_subspace* hence this subspace is appended to the list of locally relevant subspaces of these pairs. On line 6, *close_points* are used as indices, the elements at these indices of the matrix *LRS* are appended with the current base subspace.

At this point, each dimension can be included several times (in fact, each dimension can be included up to $\frac{d!}{(d-p+1)!}$ times for a pair of points if they are similar in every base subspace). Therefore, an additional step is required to remove duplicated dimensions and apply any additional processing on the locally relevant subspace (line 8). In this algorithm and following experiments, we simply remove the duplicated dimensions, but more complex strategies can be applied depending on the application. For example, to reduce the effects of noisy data that can make a dimension to be included just by chance, i.e., a dimension appears

with a much lower frequency compared to the others, a frequency cutoff can be enforced to retain only dimensions that are included frequently enough. This can also guarantee that the locally relevant subspace only contains dimensions with a certain frequency, according to Lemma 4.3.2.

The search for the base similarities and LRS is done only once and can be processed in advance. Nevertheless, it is computationally expensive as it requires traversal through all possible base subspaces and is quadratic to the volume of the input. We discuss the time complexity and provide a relaxed version of the algorithm with reduced complexity in the discussion section.

After the LRS between all pairs of points are determined, the second step of computing the similarity is straightforward, as described in Algorithm 3. Intuitively, this algorithm takes two points in the full dimensional space as input (provided that their locally relevant subspaces are already computed in the previous step). The relevant subspace is looked up on line 1, then these two points are projected onto this subspace (line 2 to line 8). Subsequently, a common distance metric (Euclidean, Cosine, Manhattan, etc.) is applied to the projections of the pair of points onto its LRS. Note that the algorithm returns both the *distance* between the pair of points, and their *locally relevant subspace*. Both values are equally important in determining the similarity between two points. For example, two points that are similar in only two dimensions and largely different in other dimensions can still result in very small distance but also only in a two-dimensional subspace. As such, comparing similarities *for a group of points* is not trivial. We show in Chapter 5 how our clustering algorithm considers both of these aspects when clustering groups of points.

4.3.5 Time Complexity

The time complexity of our algorithm is dominated by the search for the LRS of points. For a pair of points, this process traverses through all candidate base subspaces of the original data space. Depending on the selection of the value of p (the dimensionality of base subspaces), the time complexity can significantly vary. If $p = 2$, the total number of subspaces is $\frac{d(d-1)}{2}$. In general, the total number of base subspaces is $\binom{d}{p}$, hence the time complexity can be combinatorial. In each base subspace, the following steps are carried out:

- The projection of two points onto a base subspace, and the distance computation, has time complexity of $O(p)$. In practice, as p is usually low according to the definition of a base subspace, this can be considered constant time.
- The comparison of the distance with the threshold is $O(1)$.

Therefore, the search for LRS between two points has the complexity of $O(\frac{d^{p-1}}{p-1}! \times p)$, and the time complexity of Algorithm 2 becomes $O(\frac{d^{p-1}}{p-1}! \times p \times n^2)$. In practice, this process *can be performed in advance* for all pairs of points of the input.

Subsequently, the computation of the similarity between two points involves three steps:

- The algorithm looks up the LRS of two points. This takes constant time.
- It then projects the two points onto the LRS. The time complexity of this step is $O(p)$, although it can also be constant time in practice with the low value of p.
- A distance metric is then applied on the projection of the two data points. The time complexity of this step is $O(m)$, where m is the dimensionality of the LRS.

In summary, the time complexity of our similarity computation is only $O(m \times p)$, given that the LRS of the points are already computed.

It is evident that Algorithm 2 is computationally expensive. We propose the following approaches to reduce the time complexity:

- Using $p = 2$ reduces the number of base subspaces to be considered. It is arguable that setting $p = 1$ will even further reduce the time complexity. However, there are substantial advantages of having $p > 1$ that we will present in greater detail in the next section.
- Sampling base subspaces instead of checking base similarities in *all* base subspaces greatly reduces the time complexity. This comes from the observation that regardless of how many times a dimension is checked to be part of a base subspace, it is involved at most once in a LRS. Our proposal is that the algorithm just needs to check a sufficient set of base subspaces in order to determine which dimensions contribute to the formation of the LRS.

The relaxed version of the Algorithm is presented in Algorithm 4.

The sampling of base subspaces instead of an exhaustive assessment can potentially result in some dimensions being wrongly excluded from the LRS. We show in the evaluation the effects of sampling, and show that this approach can still produce similarities close to Algorithm 2 while greatly reducing the time complexity.

4.4 Evaluation

We generate synthetic datasets to evaluate the effectiveness of our proposed algorithm in comparison with other similarity measures. Let x_1 and x_2 be two points in the data space

Algorithm 4: Find locally relevant subspaces with subspace sampling

Input : $X \in \mathbb{R}^{n \times d}$
dim_min_freq: min frequency of each dimension for early termination
$sim\_threshold$

Output : $LRS_{n \times n}$: a matrix of locally relevant subspaces of each pair of points.

```
// generate d(d-1) 2-dimensional base subspace
```
$all\_2d\_base\_subspaces = combination(d, 2)$
shuffle($all\_2d\_base\_subspaces$)
```
// initialize a zero matrix to keep track of the frequency of
   each dimension
```
$dimension\_tally = O_{1 \times d}$
```
// assess point similarity in each base subspace
```
foreach *base_subspace in all_2d_base_subspaces* **do**
    $dimension\_tally_{base\_subspace} += 1$
    $projected\_X$ = project(X,$base\_subspace$)
    sim_X = point_to_point_distance($projected\_X$)
    $close\_points$ = filter(sim_X,$sim\_threshold$)
    $LRS_{close\_points} \leftarrow base\_subspace$
    // check for early termination
    if $min(dimension\_tally) > dim\_min\_freq$ **then**
        break
    end
end
$LRS = remove\_duplicated\_dim(LRS)$
return *LRS*

having $d + d'$ dimensions, in which d dimensions are relevant to the similarity while the other d' dimensions are not.

First, we evaluate the effectiveness of our algorithm with different values of d and d'. We then increase the dimensionality of our synthetic datasets up to 5,000 to evaluate the derived algorithm using the subspace sampling technique. The results are compared with our proposed ground truth distances.

4.4.1 Data Generation

We want to evaluate the ability of the algorithm to measure both similarities between points that are close (more similar) and points that are far apart. To this end, for each input of the evaluation, we generate two subsets of points, each of which contains points that are close to each other. At the same time, these two subsets of points are far from each other. At this stage of the data generation, all dimensions are relevant and contribute to the similarity

between points. To highlight the challenge of local feature relevance, and demonstrate how the irrelevant dimensions can disrupt the local similarity, we introduce irrelevant dimensions in which the points are uniformly distributed and there is no distinct grouping of points based on the similarities.

The details of the generation of a synthetic dataset are as follows:

- The first group $G1$ of close points contains 250 d-dimensional points that follow the normal distribution $N(\mu_1,\sigma_1)$.
- The second group $G2$ of close points contains 250 d-dimensional points that follow the normal distribution $N(\mu_2,\sigma_2)$. These two groups form a set of 500 d-dimensional points.
- To generate the irrelevant dimensions, we generate a matrix of size $500 \times d'$ whose values follow a uniform distribution in the range $[0, 10]$. This set of $500 \times d'$ points are concatenated horizontally to the group of close points above to form a dataset of size $500 \times (d + d')$, which is an input for our evaluation.

Figure 4.2 shows an example of a synthetic dataset with $d = 100$, $d' = 20$, $\mu_1 = 1$, $\mu_2 = 3$, $\sigma_1 = 0.1$, $\sigma_2 = 0.2$. We generate datasets with different values of d, d', μ_1, μ_2, σ_1, σ_2 to evaluate different aspects of the algorithms.

4.4.2 Evaluation of the Algorithm on Relevant Dimensions

We first vary d in the range $[20, 200]$ while keeping $d' = 20$ to evaluate the algorithm with datasets that have different numbers of relevant dimensions. In this experiment, we use $\mu_1 = 1$, $\mu_2 = 3$, $\sigma_1 = \sigma_2 = 0.1$. The parameters used in this experiment are $p = 2$, $sim\_threshold = 0.2$. To ensure the algorithm can differentiate between close points and distant points, we compute both the *intra-group* and *inter-group* point-to-point distances. The intra-group distances of $G1$ form a matrix of size 250×250 that contains the distances between every point to the other points of the group $G1$. The inter-group distances between $G1$ and $G2$ also form a matrix of size 250×250 that contains the distances between every point of group $G1$ to all points of group $G2$. For each dataset, the following distances are generated for comparison:

- The Euclidean distances between the points in the first d dimensions. This can be considered the "true" distances between the points in the relevant dimensions if the relevant dimensions are known in advance. In this evaluation, we call these the *ground truth distances*.

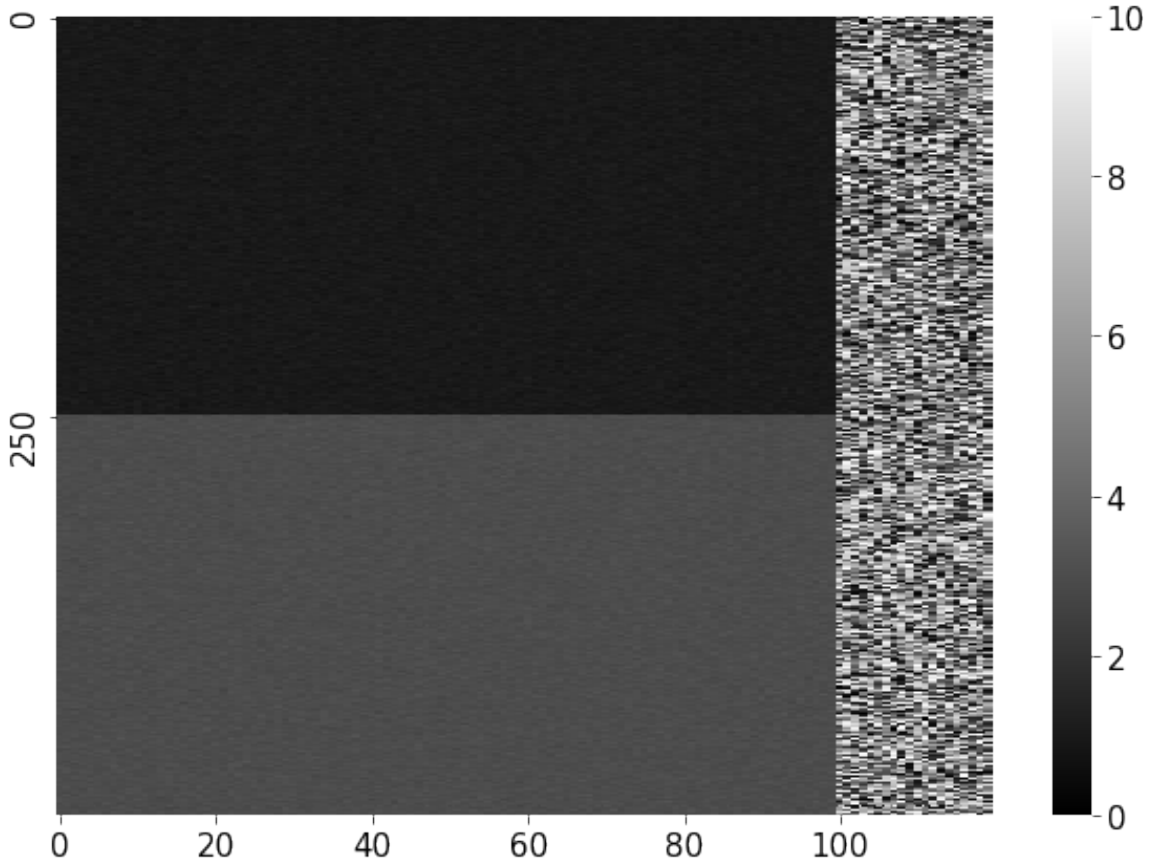


Fig. 4.2 The heatmap of a dataset used for evaluation. The dataset is generated with parameters $d = 100$, $d' = 20$, $\mu_1 = 1$, $\mu_2 = 3$, $\sigma_1 = 0.1$, $\sigma_2 = 0.2$. The values in the irrelevant dimensions $\{101...120\}$ follow a uniform distribution in the range $[0, 10]$. Each row corresponds to a point, each column corresponds to a dimension. The gray intensity of each cell reflects the value of the point in a dimension. Rows $[1, 250]$ represent group $G1$, rows $[251, 500]$ represent group $G2$.

- The Euclidean distances between the points in the entire (d+d')-dimensional data space. These are traditional distances that do not differentiate between relevant and irrelevant dimensions and can be used as the base line.
- The distance using our Local Similarity (LS) algorithm.

Table 4.1 and Figure 4.3 summarize the statistics of three types of distances we compute for different values of d. It can be observed that the traditional Euclidean distances applied in the full data space do not reflect the similarity between points. The average distances between points in the same group are distinctly very high for normal distributions with a standard deviation of 0.1, and much higher than the ground truth distances. This illustrates the challenges of the curse of dimensionality, which states that points are spread further apart

Table 4.1 The statistics of Euclidean distances, ground truth distances and LS distances in high dimensional space as d varies in the range $[20, 200]$, $d' = 20$.

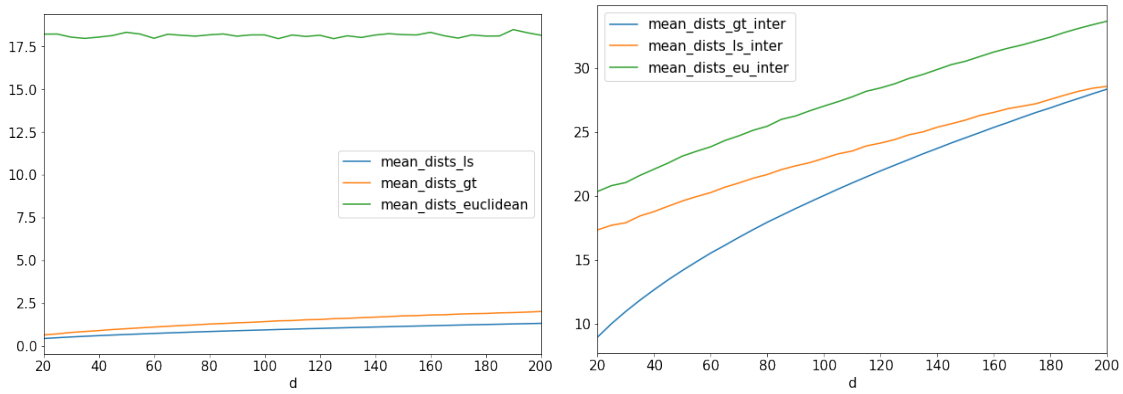
	Intra-group distances							
	Euclidean distances		Ground truth distances		LS distances		LS distances (with sampling)	
d	(min,max)	mean ± std	(min,max)	mean ± std	(min,max)	mean ± std	(min,max)	mean ± std
20	(8.25,27.22)	18.21±2.39	(0.26,1.07)	0.63±0.10	(0.17,0.63)	0.42±0.06	(0.22,1.33)	0.32 ±0.34
50	(7.71,27.90)	18.32±2.43	(0.63,1.42)	0.99±0.11	(0.38,0.88)	0.66±0.07	(0.31,0.71)	0.50 ±0.16
100	(8.55,27.75)	18.17±2.44	(1.03,1.83)	1.41±0.09	(0.79,1.18)	0.93±0.07	(0.60,1.00)	0.74 ±0.17
150	(7.47,27.45)	18.18±2.44	(1.34,2.18)	1.74±0.10	(0.98,1.40)	1.13±0.09	(0.69,1.14)	0.92 ±0.28
200	(7.80,26.70)	18.15±2.46	(1.55,2.41)	2.00±0.09	(1.01,1.54)	1.31±0.1	(0.97,1.26)	1.06 ±0.15
	Inter-group distances							
	Euclidean distances		Ground truth distances		LS distances		LS distances (with sampling)	
d	(min,max)	mean ± std	(min,max)	mean ± std	(min,max)	mean ± std	(min,max)	mean ± std
20	(11.81,28.83)	20.36±2.14	(8.42,9.47)	8.97±0.13	(3.12,28.83)	17.34±1.09	(4.27,28.83)	20.00 ± 3.24
50	(16.27,31.62)	23.12±1.93	(13.54,14.67)	14.18±0.14	(7.51,31.62)	19.61±1.22	(8.29,31.62	22.74 ± 3.13
100	(21.43,34.56)	27.03±1.65	(19.48,20.56)	20.04±0.14	(8.27,34.56)	22.95±1.41	(10.53,34.75)	27.02 ± 1.84
150	(25.76,36.91)	30.54±1.43	(24.02,25.11)	24.55±0.14	(14,23,36.91)	25.94±1.51	(17.52,36.91)	30.50 ± 2.15
200	(29.06,39.54)	33.67±1.32	(27.79,28.84)	28.36±0.15	(15.95,39.54)	28.57±1.68	(18.36,39.54)	33.58 ± 2.38

as the number of dimensions increases. To this end, the similarity between points can easily be obscured by a few irrelevant dimensions. Here, although only 20 out of 200 dimensions are irrelevant (last row of Table 4.1), the distances can already be changed significantly.

The results also show that our proposed algorithm can find the similarities between points. The close points are distinguished from the further points, reflected in small intra-group distances compared to significantly higher inter-group distances. Figure 4.3a also shows that the *intra-group* LS distances also track the ground truth distances closely. These two distances have a Pearson correlation of 0.99, indicating that the relevant dimensions are correctly picked up by the algorithm to be included in the LRS. Moreover, Figure 4.3b shows that although the *inter-group* LS distances are distinctive from the ground truth distances when the proportion of irrelevant dimensions is high (e.g., when $d = d' = 20$), the gap between two distances shrinks as the proportion decreases. In fact, when $d = 200$, $d' = 20$, the LS distance is 28.57 compared to the ground truth distance of 27.79.

In addition to the distances, we also evaluate the ability of our algorithm to find the LRS of each pair of points using the prior knowledge of d relevant dimensions as ground truth. We consider a *true positive* as a relevant dimension that is correctly picked up by the algorithm to be included in a LRS and a *false negative* as a relevant dimension that was not included in a LRS. We focus on two main metrics as follows and summarize the results in Table 4.2a:

- The *true positive rate (recall)* indicates how often the algorithm correctly includes a relevant dimension into the LRS.
- The *false negative rate* indicates how often the algorithm incorrectly excludes relevant dimensions from a LRS.



(a) (b)

Fig. 4.3 The average values of Euclidean distances, ground truth distances, and LS distances as d varies in the range [20,200] and $d' = 20$ for (a) close points in group $G1$ (intra-group distances), and (b) between points of groups $G1$ and $G2$ (inter-group distances).

Table 4.2 Evaluation of dimensions of the LRS: (a) Original LS algorithm, (b) LS algorithm with sampling

d	20	50	100	150	200
FNR	0.56	0.38	0.29	0.25	0.23
TPR	0.44	0.62	0.71	0.74	0.77
F1 Score	0.61	0.76	0.83	0.86	0.87

(a)

d	20	50	100	150	200
FNR	0.70	0.58	0.48	0.43	0.40
TPR	0.30	0.41	0.51	0.56	0.59
F1 Score	0.45	0.58	0.68	0.72	0.75

(b)

Table 4.2a shows that as the proportion of irrelevant dimensions is high (e.g., $d = d' = 20$), the algorithm is only able to correctly identify around 50% of relevant dimensions. However, this is improved as d grows larger than d'. By not having all the correct dimensions included in the LRS, the LS distances are affected in the following ways:

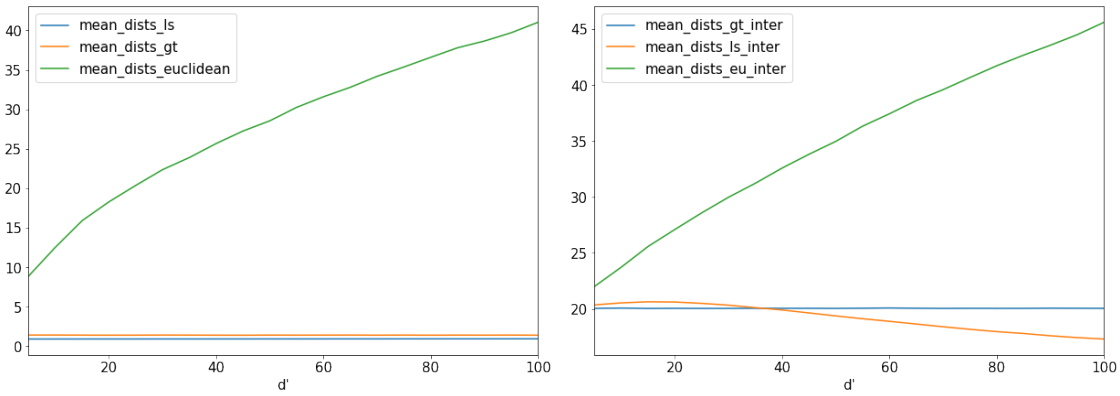
- The lack of inclusion of relevant dimensions to the LRS results in lower dimensional LRS, which subsequently leads to LS distances being smaller than the ground truth intra-group distances.
- While computing LS distances for inter-group points that are far from each other, in addition to all the d relevant dimensions being correctly identified, the inclusion of some irrelevant dimensions (in which the points are still far) further amplifies the dissimilarities between points.
- The maximum values of LS and Euclidean for inter-group distances are similar. These represent the pairs of points for which our algorithm fails to find the LRS, hence the full dimensional space is assumed when computing distances.

4.4.3 Evaluation of the Effects of Irrelevant Dimensions on the Algorithm

We fix d to 100 while varying d' in the range $[5, 100]$ in this experiment. All other values of μ_1, μ_2, σ_1, σ_2 remain the same as the values used in the previous experiment. All parameters of p and *sim_threshold* are also reused from the previous experiment.

Table 4.3 The statistics of Euclidean distances, ground truth distances and LS distances in high dimensional space as d' varies in range $[5, 100]$, $d = 100$.

	Intra-group distances					
	Euclidean distances		Ground truth distances		LS distances	
	(min,max)	mean ± std	(min,max)	mean ± std	(min,max)	mean ± std
d'=5	(1.53,18.55)	8.87±2.46	(1.06,1.85)	1.42±0.10	(0.73,1.16)	0.92±0.08
d'=25	(9.95,30.83)	20.37±2.50	(1.01,1.81)	1.41±0.1	(0.89,1.15)	0.93±0.08
d'=50	(19.98,38.20)	28.55±2.43	(1.00,1.85)	1.42±0.1	(0.90,1.18)	0.93±0.08
d'=75	(25.06,43.74)	35.36±2.38	(1.03,1.86)	1.42±0.10	(0.98,1.17)	0.94±0.09
d'=100	(30.05,51.41)	41.02±2.44	(0.99,1.77)	1.40±0.1	(1.05,1.20)	0.95±0.08
	Inter-group distances					
	Euclidean distances		Ground truth distances		LS distances	
	(min,max)	mean ± std	(min,max)	mean ± std	(min,max)	mean ± std
d'=5	(19.68,26.62)	22.00±2.46	(19.54,20.65)	20.05±0.13	(9.27,26.62)	20.35±2.50
d'=25	(23.00,37.10)	28.58±2.50	(19.45,20.73)	20.04±0.14	(10.02,37.10)	20.49±11.90
d'=50	(26.11,42.77)	34.97±2.43	(19.38,20.60)	20.04±0.14	(10.15,42.77)	19.37±17.57
d'=75	(32.72,49.61)	40.67±2.38	(19.50,20.63)	20.04±0.14	(11.71,49.61)	18.17±17.99
d'=100	(36.31,55.32)	45.61±2.44	(19.52,20.65)	20.05±0.15	(13.96,55.32)	17.31±15.83



(a) (b)

Fig. 4.4 The average values of Euclidean distances, ground truth distances, and LS distances as d' varies in range [5,100] and $d = 100$ for (a) close points in group $G1$ (intra-group distances), and (b) between points of groups $G1$ and $G2$ (inter-group distances).

The results show that the LS distances do not increase, and have a high Pearson correlation of 0.99 with the ground truth distances. This proves that the algorithm is able to identify only relevant dimensions and does not include irrelevant dimensions, despite its high proportion. Here, the maximum proportion of irrelevant dimensions is 100% when $d = d' = 100$.

The results also show that the Euclidean distances applied in the full dimensional space cannot distinguish between points that are near to each other from points that are far from each other. Figure 4.4a and 4.4b shows that the intra-group average distances and inter-group average distances converge as the number of irrelevant dimensions d' increases. This illustrates the phenomenon that points are spread to become equidistant due to noise in the irrelevant dimensions as the number of dimensions increases [12].

Figure 4.4b and Table 4.3 also shows the non-homogeneous changes and the large standard deviations of the inter-group LS distances as d' increases. With higher values of d', there are more instances in which two points in two different groups are near by chance in a certain irrelevant base subspace; therefore this low dimensional base subspace is assumed to be the LRS of these two points, resulting in very small distances compared to the distance if it is computed in the full dimensional space otherwise. The higher rate of occurrence of these instances results in more low inter-group LS distances, which not only increase the standard deviations but also decrease the overall average distances. One potential method to overcome this issue is to have a more complex method of *remove_duplicated_dim* (line 8 of Algorithm 2); for example, a simple rule that enforces a dimension to be present at least x times to be included in the LRS can fix the issue above.

In this experiment, the FNR and TPR are recorded as a constant of 0.23 and 0.77 respectively as d' increases from 5 to 100. These numbers show that the rate that a dimension is correctly selected for the LRS is not affected by the irrelevant dimensions, i.e., as $d = 100$, only 23 dimensions are incorrectly not included in the LRS regardless of the number of noisy and irrelevant dimensions.

4.4.4 Evaluation of the Algorithm with Sampling Technique

We evaluate the proposed algorithm that uses the sampling technique (Algorithm 4) using datasets having much higher dimensionalities in the range of [20,5000]. We keep the number of irrelevant dimensions $d' = 20$ for $d < 200$ to provide a benchmark with the results of the original algorithm discussed above. For higher values of d, we set $d' = d \times 0.1$. In order to keep the algorithm linear to the dimensionality, we perform sampling such that each dimension is checked $\lceil \sqrt{d+d'} \rceil$ times. The time complexity of the LRS generation hence becomes $O((d+d') \times p \times n^2)$

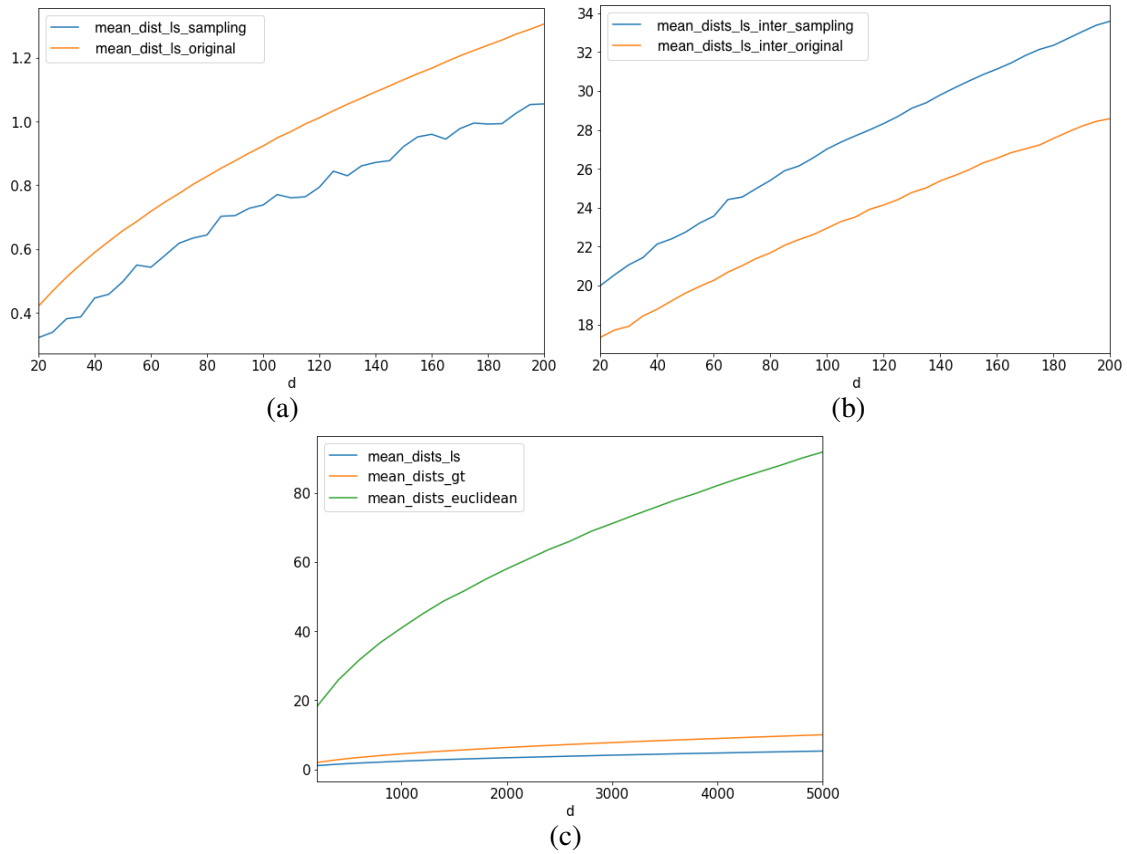


Fig. 4.5 Evaluation results of the algorithm with sampling: (a) Comparison of intra-group LS distances, (b) Comparison of inter-group LS distances, (c) Intra-group LS distances as d varies in range $[200, 5000]$ and $d' = 0.1 \times d$.

The LS distances generated by the algorithm with the sampling approach, which will be referred to as $LS_{sampling}$ onwards, are summarized in the last column of Table 4.1 and illustrated in Figure 4.5. When $d \leq 200$, the $LS_{sampling}$ is highly correlated with the LS distances generated by the original ($LS_{original}$) with a Pearson correlation of 0.97. The results show that $LS_{sampling}$ fluctuates more and has larger standard deviations. This is expected since the dimensions are sampled and not exhaustively checked with every other dimension, which leads to more dimensions being not correctly included in the LRS. We also observed from the results that there are more false negatives (a relevant dimension is not included) when computing distances for points that are near, resulting in $LS_{sampling}$ being smaller than $LS_{original}$. Besides, there are more false positives when computing distances for points that are far away, resulting in $LS_{sampling}$ being larger than $LS_{original}$ for inter-group distances. The evaluation of the dimensions of LRS are summarized in Table 4.2b

Table 4.4 A set of five points in 5-dimensional space.

	d_1	d_2	d_3	d_4	d_5
x_1	1	1	1	1	1
x_2	1	1	1	1	10
x_3	4	1	5	1	2
x_4	1	6	3	2	5
x_5	2	1	1	1	7

As the number of dimensions grows to 5000 (and the number of irrelevant dimensions grows to 500), the algorithm is still able to correctly find the small distances between similar points that are near in the same group and identify points that are far in the other group. Figure 4.5c shows that the $LS_{sampling}$ grows at a much slower rate compared to the Euclidean distances, and is close to the ground truth distances.

4.5 Discussion

We start the discussion of our proposed method with a simplified example to demonstrate how it overcomes the challenge of local feature relevance to produce better similarity measures. Consider a set of 5 points x_1, x_2, x_3, x_4, x_5 in a 5-dimensional space as presented in Table 4.4. We assess different methods to measure similarities between points in the context of finding subspace clusters.

Table 4.5 Distance matrices

	x_1	x_2	x_3	x_4	x_5
x_1	0	9.0	5.1	6.8	6.1
x_2	9.0	0	9.4	7.4	3.2
x_3	5.1	9.4	0	6.9	6.7
x_4	6.8	7.4	6.9	0	5.9
x_5	6.1	3.2	6.7	5.9	0

(a) Euclidean distance

	x_1	x_2	x_3	x_4	x_5
x_1	0	9	8	12	7
x_2	9	0	15	13	4
x_3	8	15	0	14	11
x_4	12	13	14	0	11
x_5	7	4	11	11	0

(b) Manhattan distance

It can be observed from Table 4.5 that Euclidean and Manhattan distances are evidently not an effective measurement. Specifically, point x_1 is deemed to be more similar to point x_3 than to point x_2, i.e., $d_{euclidean}(x_1,x_3) = 5.1 < d_{euclidean}(x_1,x_2) = 9.0$ and $d_{manhattan}(x_1,x_3) = 8.0 < d_{manhattan}(x_1,x_2) = 9.0$. This single number metric also obfuscates the fact that two points x_1, x_2 are identical in the subspace d_1, d_2, d_3, d_4. Note that our proposed algorithm also discovers this information by returning the locally relevant subspaces. It is the irrelevant

	x_1	x_2	x_5
x_1	0	0	1
x_2	0	0	1
x_5	1	1	0

Table 4.6 Meaningful distances between x_1, x_2, x_5 in the relevant subspace.

dimension that disrupts the similarity between x_1 and x_2 in the relevant subspace as we highlighted in the Evaluation section.

Applying PCA or MDS to reduce the dimensionality and visualise the data in a two-dimensional space does not yield good results either. Figure 4.6 shows that all the points are well separated from each other and therefore implies that no meaningful proximity or cluster tendency exists in the given dataset.

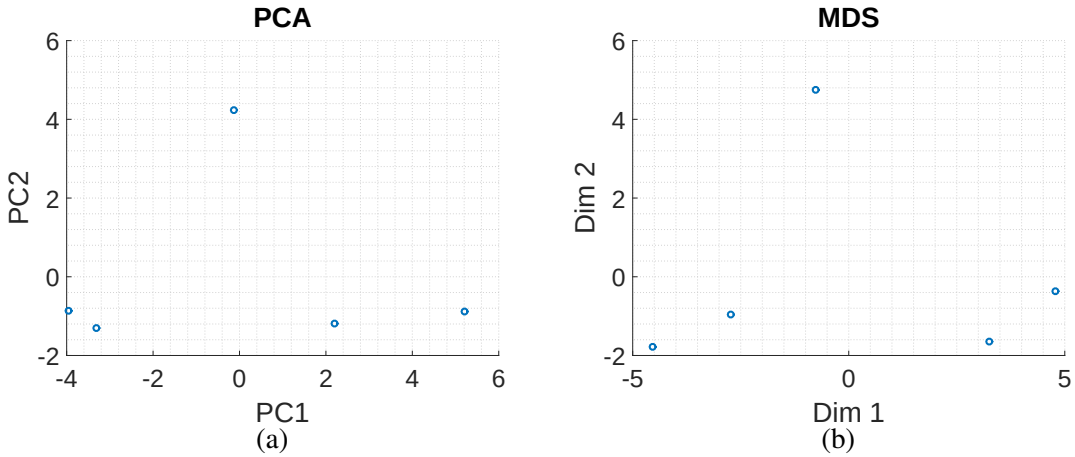


Fig. 4.6 Applying dimensionality reduction techniques on the data.

Our proposed algorithm considers the proximity between points in each 2-dimensional subspace. It then detects that points x_1, x_2, x_5 are consistently near each other in subspaces $\{d_1, d_2\}$, $\{d_1, d_3\}$, $\{d_1, d_4\}$, $\{d_2, d_3\}$, $\{d_2, d_4\}$, $\{d_3, d_4\}$. These $2D$ subspaces constitute a $4D$ subspace (each dimension is involved 3 times). The final output of our algorithm indicates the relevant dimensions between points x_1, x_2, x_5 are d_1, d_2, d_3, d_4 and the similarities between these points in the relevant subspace are summarized in Table 4.6.

From this example, it can be argued that checking for base similarities between points in each individual dimension (i.e., $p = 1$) can also discover the local feature relevance and achieve the same similarity in subspace $\{d_1, d_2, d_3, d_4\}$ for lower computational complexity. We further justify our proposed method in the next section and present how it produces superior results. We then explain the reason why searching for base similarities in $2D$

subspaces is a balance between computational complexity and the quality of similarity measures.

4.5.1 Preservation of Covariances Between Dimensions

Not only is the proximity between points in each dimension important, but the covariances of values in different dimensions are also critical to the formation of clusters in multi-dimensional space. We demonstrate that our proposed method of measuring similarities is able to preserve this attribute and subsequently improve clustering in different ways.

First, it eliminates at an early stage the unimportant similarities in lower dimensional subspaces that are not applicable to cluster formation. To this end, the computational efficiency is improved. Figure 4.8a shows a distribution of 300 points in 3-dimensional space. Points $\{x_i\}_{i=1}^{100}$ follow a normal distribution $N(1,2)$ and form a dense unit in dimension d_1, and are uniformly distributed in dimensions d_2, d_3. Similarly, $\{x_i\}_{i=101}^{200}$ and $\{x_i\}_{i=201}^{300}$ follow two normal distributions $N(7,2)$ and $N(10,2)$ in d_2 and d_3, and form two dense units in these dimensions. When clustering these points in $2D$ and $3D$ spaces, where covariance is implicitly implied, these points do not form any cluster, as confirmed by k-means or visual inspection of Figure 4.8a. This can be explained with the normal probability density distribution in Figure 4.7b. While the first 100 points $\{x_i\}_{i=1}^{100}$ are close to each other in d_1, the same set of points have large variances in d_2 and d_3, and cannot be considered close in higher dimensional space. The correlation between different dimensions is omitted when each dimension is considered separately. This example demonstrates the common cases where similarities in individual dimensions do not always result in similarities in high dimensional space; therefore *efficiency could be improved by eliminating these redundant similarities at early stages*. In fact, the abundant checks explain the low scalability of most of the bottom-up clustering algorithms to the number of dimensions [160]. These algorithms start the search for similar points (dense units) in each individual dimension before aggregating them to form higher dimensional clusters. This leads to high computational complexity, e.g., if there are on average m dense units in each dimension, each dense unit needs to be checked with other $(d-1) \times m$ dense units in other $d-1$ dimensions. Hence, the first level of aggregation (to combine one-dimensional dense units to two-dimensional clusters) would need to check $O(m^d)$ pairwise possible aggregations, where d is the number of dimensions.

Second, taking covariances into account discovers similarities in high dimensional space that are otherwise undetected by using other methods or by only looking for similarities in each dimension separately. Figure 4.7c shows an example where missing clusters could be prevented. It contains 300 points whose coordinates in each dimension are sampled from three equal size normal distributions $N(1,2)$, $N(7,2)$ and $N(10,2)$. In fact, if we consider

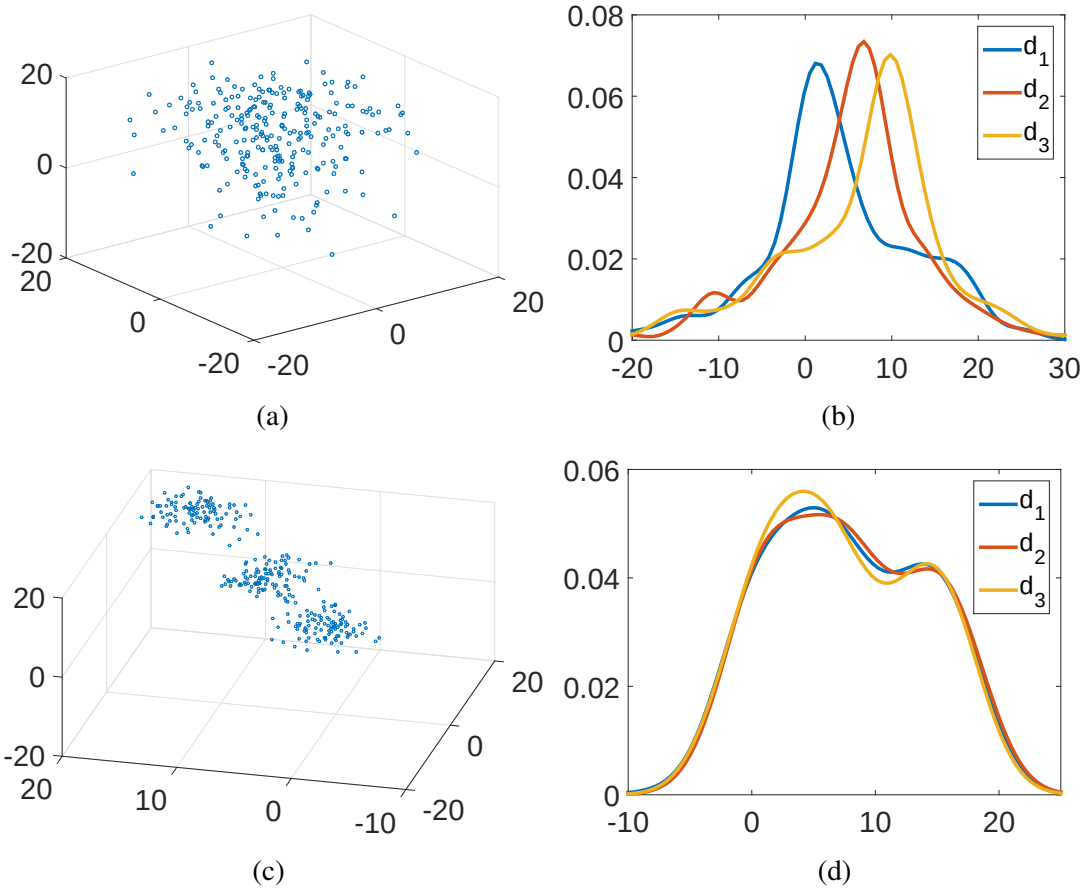


Fig. 4.7 (a) Distribution of 300 points in 3-dimensional space; (b) Estimated normal probability density of 200 points in each dimension; (c) Distribution of 300 points having the same distribution as (a) in each dimension, but with small covariances; (d) Estimated normal probability density of (c).

each dimension separately, the values in each dimension are the same as the previous distribution shown in Figure 4.8a. However, in this example, we enforce that for each point x_i, its coordinates in all dimensions must be drawn from the same distribution. No dense units are found in each individual dimension as can be observed from the probability density distribution in Figure 4.7d (which do not show any significant peaks, compared to Figure 4.7b). With no dense unit in each dimension, the similarity is not further aggregated in higher dimensional spaces. However, it is visually evident that three clusters of points that are more densely distributed exist in this dataset.

In summary, we show that searching for base similarities in subspaces higher than one-dimension preserves the covariances of values between different dimensions, and subsequently not only improves the efficiency by eliminating redundant checks, but also detects similarities in high dimensional spaces that are otherwise missed by existing methods.

4.5.2 Selection of parameters p

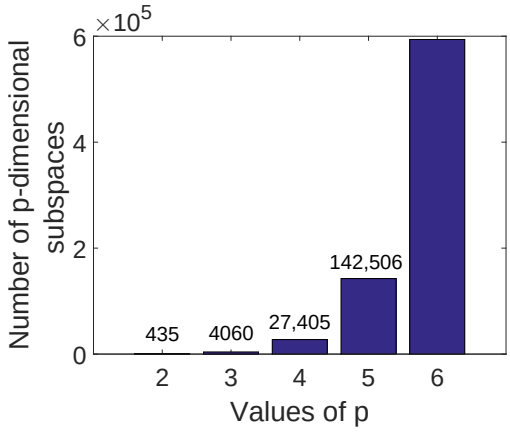
p=1

We presented in the previous section the benefits of learning the covariances of coordinates in different dimensions. Specifically, we first argue that by checking similarities in 2-dimensional base subspaces instead of in each dimension individually, although our algorithm increases the number of base subspaces to be checked from $O(d)$ to $O(d^2)$, the overall complexity of the algorithm is not exponential to the dimensionality of d and it is still able to learn covariances between dimensions. Note that the overall time complexity can be further reduced as we demonstrated our sampling approach in the Evaluation section. Second, by assessing covariances of data points in different dimensions, the algorithm can detect similarities in high dimensional spaces that are overlooked in individual dimensions. For these reasons, setting $p = 2$ is the minimum value of p that allows more than one dimension to be assessed simultaneously in order to preserve covariances between dimensions.

p>2

Searching for base similarities in subspaces of p dimensions where $p > 2$ (i.e., $p = 3, 4, 5, etc.$) also considers the covariances of values in different dimensions. However, doing so can significantly increase the computational complexity. Specifically, the total number of p-dimensional subspaces in a d-dimensional data space is $\binom{n}{p}$. This number grows in a combinatorial order with respect to p. Figure 4.8 illustrates the rapid growth of the number of subspaces with respect to different values of p in a 30-dimensional dataset. It can be observed that the problem quickly becomes intractable for a fairly low dimensional dataset.

Although this high complexity can be reduced by using our sampling approach, choosing high values of p can lead to "true" similarities being overlooked. Specifically, if the method starts the search for base similarities in p-dimensional subspaces, it assumes that the spaces where meaningful similarities exist have at least p dimensions. Consequently, the method misses all valid base similarities in lower dimensional subspaces and subsequently their aggregation into meaningful similarities in higher dimensional space. For example, if p is set to 3, all valid similarities in $2D$ subspaces are ignored, and all higher dimensional similarities being aggregated from these base similarities are not detected. In reality, those $2D$ base similarities could be aggregated to form similarities in $4D$ subspaces. Moreover, if p is set to a value higher than 2, it is possible that the search for similarity in base subspaces already suffers from the local feature relevance challenge, that is, not all dimensions in the base subspaces are guaranteed to be relevant. For this reason, it is ideal to start the search for base similarities in low dimensional subspaces, i.e., to keep p small.



(a)

Fig. 4.8 Number of p-dimensional subspaces.

4.6 Summary

In this chapter, we proposed a novel method to measure similarities that can overcome local feature relevance in high dimensional space. The algorithm relies on our proposed approach that the similarities between data points in high dimensional space can be reflected by their similarities in the component base subspaces. This approach is proven in Lemmas 4.3.1, 4.3.1, 4.3.3, and 4.3.4. To this end, the algorithm first assesses similarities between points in low dimensional base subspaces in order to determine the relevant subspace for each pair of points, before applying a traditional measurement in that subspace, whose dimensions are all guaranteed to be relevant. The evaluation shows that our proposed algorithm is effective in determining the locally relevant subspace, finding the similarities between points, and distinguishing between points that are close from points that are further away. We show that the proposed algorithm is not affected by the noisy data in irrelevant dimensions.

We focus on base similarities in $2D$ subspaces as this configuration provides a balance between computational complexity and the quality of the measurements. It allows the algorithm to implicitly check the covariances of values between different dimensions while still keeping the computational complexity low. However, depending on the available resources and attributes of the data, our proposed algorithm has the flexibility to search for base similarities in higher dimensional subspaces.

The complexity analysis shows that the proposed algorithm has high time complexity and it grows quadratically with the number of dimensions as well as the volume of the input. To this end, we propose a derived version of the algorithm using a sampling technique and show that the time complexity can be reduced to be linear to the dimensionality while still

achieving comparable results. Nevertheless, the algorithm is evidently still computationally expensive, which makes it unfit for the task of clustering large volumes of high dimensional data.

The bottleneck of the algorithm is in the search for LRS for all pairs of points. However, this task is only necessary if the exact distances between points are expected. All of our proposed theory on aggregating base similarities to form similarities in higher dimensional spaces (Lemmas 4.3.1-4.3.4) is still valid *without computing the exact similarities*. For the purpose of clustering, only the grouping of similarities between points is required. To this end, we integrate our proposed approach of computing similarities in high dimensional space into the application of clustering and propose an efficient subspace clustering algorithm in the next chapter.

Chapter 5

Scalable Bottom-up Subspace Clustering using FP-Trees for High Dimensional Data

Subspace clustering aims to find groups of similar objects (clusters) that exist in lower dimensional subspaces from a high dimensional dataset. One of the fundamental challenges of subspace clustering is to measure similarities between data points, for which we have proposed a similarity measurement and demonstrated its effectiveness in finding point-to-point similarities in high dimensional space. However, it is not trivial to apply this measurement in its form to groups of points in the context of clustering. We propose a two-phase algorithm Subspace Clustering by aggregation of Base Similarities (SCBS) that integrates our proposed similarity measurement to effectively find clusters in high dimensional space. The algorithm first searches for base similarities by performing clustering in low dimensional base subspaces to find base clusters. It then constructs clusters in higher-dimensional subspaces from the base clusters, which we transform to a frequent pattern mining problem. This transformation enables efficient search for clusters in higher-dimensional subspaces, which is done using FP-Trees. By taking this approach, our proposed clustering algorithm is able to find clusters in non-disjoint subspaces, which remains a challenge to many state-of-the-art subspace clustering algorithms, and is able to scale to large volume datasets of up to 1 million data points. We also analyze the limitations of the algorithm in scaling to high dimensionalities, and subsequently propose a derived clustering algorithm using a subspace sampling approach that can cluster a reasonably large volume dataset having up to 3,500 dimensions. We thoroughly evaluate SCBS with other subspace clustering algorithms using synthetic datasets that have different volumes, dimensionalities and levels of outliers. The empirical results show that the proposed algorithm produces good

clustering qualities and can scale to the volumes and dimensionalities of datasets produced by real life applications in the field of the Internet of Things and bioinformatics.

5.1 Introduction

Subspace clustering aims to find groups of similar objects, or clusters, that exist in lower dimensional subspaces from a high dimensional dataset. This has a wide range of applications, including the rapidly growing fields of the Internet of Things (IoT) [115] and bioinformatics [50]. The proliferation and applications of subspace clustering are discussed in greater detail in the introduction chapter of this thesis. One of the fundamental challenges of subspace clustering, or more generally in analyzing high dimensional data, is to overcome the problem of local feature relevance, which limits the effectiveness of traditional distance metrics.

We show in Chapter 4 how our proposed similarity measure can capture the point-to-point similarities effectively in high dimensional space. However it is not trivial to incorporate this algorithm in a clustering application for two main reasons:

- First, the search for the locally relevant subspaces and the subsequent computation of similarities for *all pairs of points* are both computationally expensive. A trivial approach is to compute the similarity matrix using this measurement, then subsequently apply a distance based clustering algorithm on that matrix. However, this approach makes the algorithm complexity quadratic to both the number of points and dimensions, therefore making the clustering algorithm less flexible to large datasets generated by real life applications.

- Second, the similarity depends on the subspace-context, i.e., the similarity value needs to be coupled with the subspace in which it is computed. To this end, even with unlimited resources that can support the computation of a large similarity matrix, it is not valid to just cluster points based on their numerical similarity values since they might not share a common subspace in which the similarities are assessed. To this end, any grouping of points needs to take into account both their similarities and their subspace.

The main content of this chapter has been published in the following paper:
M. T. Doan, J. Qi, S. Rajasegarar, and C. Leckie. Scalable Bottom-up Subspace Clustering using FP-Trees for High Dimensional Data. Proceedings of the 6^{th} IEEE International Conference on Big Data (IEEE BigData), 2018

In this chapter we propose a subspace clustering algorithm that leverages our proposed similarity measurement and integrates it in the clustering context. Our proposed algorithm addresses both challenges discussed above.

We take into consideration two other challenges that are commonly shared by subspace clustering algorithms. The first challenge lies in handling large inputs. This is essential for many applications nowadays since the captured data can grow to millions of records in a short period of time. The scalability challenge also applies to high numbers of dimensions. For example, sensor data captured by IoT applications or gene expression data in biomedical studies can have thousands of dimensions [7, 50]. It has been shown [160, 222] that many existing algorithms have high computational costs and take considerable time to cluster relatively small inputs, e.g., the algorithm STATPC runs in several days to cluster the Pendigits dataset (7,500 records of 16 dimensions), SUBCLU needs more than 16 hours to cluster the Diabetes dataset (768 records of 8 dimensions) [160]. Table 5.1 further shows the running time of more recent algorithms on more powerful hardware resources. It illustrates how our algorithm can scale to inputs with large volumes of data, in comparison to state-of-the-art subspace clustering algorithms SWCC [50], SSC [76], and LRR [139]. These are results that are extracted from our evaluation with synthetic datasets that have 10 dimensions and volumes of up to 100,000 data points. The running time of our algorithm over 100,000 data points is half that required by SWCC (which is a highly efficient co-clustering algorithm, but cannot find clusters in non-disjoint subspaces). The state-of-the-art subspace clustering algorithms SSC and LRR also suffer as the number of data points increases. SSC triggers memory errors when the number of data points reaches 15,000, while LRR cannot successfully terminate in 12 hours for just 5,000 points. While this example focuses on the scalability of the algorithm to the volume of the input, we present various evaluations using datasets that have different properties and much higher dimensions, together with more details of the generation process in Section 5.4.

	5,000	10,000	15,000	20,000	50,000	100,000
SCBS	**6.7**	**13.2**	**20.7**	**28.7**	**127.9**	**184.5**
SWCC	9.8	19.9	37.8	93.94	198.96	374.48
SSC	226.1	416.9	1506.4	-	-	-
LRR	-	-	-	-	-	-

Table 5.1 Clustering time (in seconds) on 10-dimensional datasets. The volume ranges from 5,000 to 100,000 points. Experimental settings are described in more detail in Section 5.4.

The second challenge involves finding clusters in non-disjoint subspaces [230]. Many recent algorithms [76, 139] assume that clusters are located in disjoint subspaces, which do not have any intersection except for the origin. This is a strong assumption that can

be unrealistic, because real-life data may be correlated in different overlapping subsets of dimensions, also known as the property of *local feature relevance* [128]. For example, with gene expression data, a particular gene can be involved in multiple genetic pathways, which can result in different symptoms among different sets of patients [71]. Hence, a gene can belong to different clusters that have dimensions in common while differing in other dimensions [98]. Figure 5.1 presents another illustrative example of clusters in non-disjoint subspaces that are observed in data collected from IoT applications, which was first introduced in Section 1.1. The heatmap visualizes the subspace clustering results of a car parking occupancy dataset at 10 locations from 9am to 1pm, where each column represents a car parking bay, and each row represents an hour of the day. Hence, each cell contains the data observed at a certain location during a specific time of the day. Each cluster is color coded to highlight the subsets of data points and subsets of dimensions that are simultaneously grouped together. It can be observed that clusters C_1 and C_2 are in non-disjoint subspaces since they share the dimensions of parking bays $P2$ and $P3$ in common. In the case of C_1, this can be interpreted as the utilization of these two parking bays following some pattern that is also observed at `P1` between 9am-10am. On the other hand, cluster C_2 shows that $P2$ and $P3$ follow a different pattern between 11am-1pm, and share that pattern with $P4$ and $P5$. Further analysis of the data can suggest that $\{P1, P2, P3\}$ are busy parking bays during morning peaks, whereas $\{P2, P3, P4, P5\}$ have higher occupancy levels during lunch time.

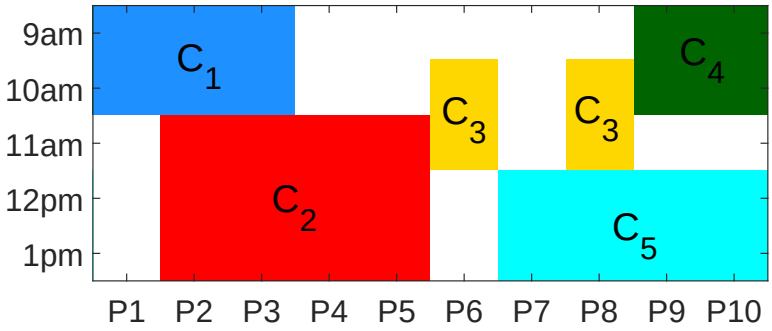


Fig. 5.1 An illustration of clusters in non-disjoint subspaces for car parking occupancy data. Clusters are highlighted to show simultaneous groupings of points and dimensions.

In order to integrate our similarity measurement and to address the challenges of scalability and clustering data in non-disjoint subspaces, we propose a two-phase algorithm to perform Subspace Clustering by aggregating Base Similarity (SCBS). First, it searches for base similarities between data points by performing clustering in low dimensional base subspaces, which we call *base clusters*. We start with base clusters instead of dense units in separate dimensions, which are used in existing bottom-up clustering algorithms [128, 89]. This allows our algorithm to preserve the covariance of data between different dimensions,

which is also a critical factor when clustering high dimensional data, as we discuss and elaborate in Section 4.5.1.

In the second phase, base clusters that cover similar sets of data points are aggregated to form clusters in higher dimensional subspaces. This process of aggregation is non-trivial. One of the main challenges lies in keeping the number of aggregated clusters tractable. This not only directly affects the computational costs of the algorithm, but also ensures that the final result is presented in an appropriate number of clusters. Many existing algorithms [14, 167] depend on combinatorial search to combine low dimensional clusters (dense units). For example, the running time of CLIQUE to sequentially find all k-dimensional dense units is exponential to k. Although pruning of redundant subspaces is performed to further increase efficiency, evaluations have shown that the time complexity is still exponential to the dimensionality of clusters [14]. We alleviate this heavy computation by transforming the greedy aggregation problem into a frequent pattern mining problem [97] to achieve efficient and robust aggregation of base clusters. This approach also allows us to avoid the construction of a similarity matrix, which has quadratic complexity with respect to the input volume. Therefore, we reduce both time and space complexity and enable the algorithm to work with inputs that have very large volumes. Despite its scalability to large volumes, SCBS does not scale well with the number of dimensions, as we show in Section 5.3.3. We overcome this challenge by proposing a derived version of the algorithm based on a subspace sampling technique, to reduce the time complexity to sub-quadratic with respect to the dimensionality. Our proof of concepts (in the previous chapter) guarantee the similarities in high dimensional space and our empirical results show that the algorithm achieves comparable results with considerably shorter running times. This allows our algorithm to cluster a reasonably large dataset with up to 3,500 dimensions within a reasonable running time.

During the aggregation process, a base cluster may be aggregated into multiple clusters in different higher dimensional subspaces that have overlapping dimensions, which enables us to find clusters in non-disjoint subspaces. The general steps of our algorithms are summarized in Figure 5.2 and are detailed in Section 5.3.

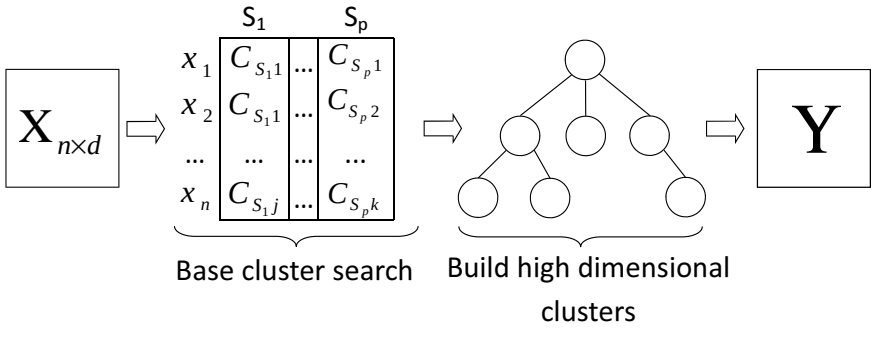


Fig. 5.2 Framework of the proposed algorithm

We make the following contributions:

- We propose a novel subspace clustering algorithm that can find clusters in non-disjoint subspaces and scale to large inputs having up to one million data points. The novelty of our approach is reflected in both phases of the algorithm. First, we search for base clusters in low dimensional subspaces to preserve the covariance of data between different dimensions. Second, we transform the process of sequential aggregation of base clusters to a problem of frequent pattern mining to construct high dimensional clusters.
- We propose a solution based on a subspace sampling approach that improves the scalability of our algorithm to high numbers of dimensions. Our sampling approach guarantees that all dimensions are sufficiently represented, and is supported by our similarity measurement.
- We demonstrate that the proposed algorithms outperform traditional subspace clustering algorithms using bottom-up strategies, as well as state-of-the-art algorithms with other clustering strategies, in terms of accuracy and scalability on data having large volumes and high dimensionalities.

5.2 Methodology

The notation used in this chapter was first presented in Chapter 2. We briefly mention the problem statement of subspace clustering before presenting the methodology of our algorithm.

Let $X = \{x_i \in \mathbb{R}^d : i = 1..n\}$ be a set of n points in a d-dimensional space, and X_j be a subset of X. *Our subspace clustering algorithm finds all clusters by identifying their corresponding subspaces and point sets*. The set of all subspace clusters is denoted as $Y = \{C_{S_i}^{X_j},\ i : 1..s,\ j : 1..c\}$. Here, s denotes the number of subspaces containing clusters, and c denotes the number of all clusters. Our algorithm aims to find clusters that reside in overlapping subspaces, hence $\forall q, r, S_q \cap S_r \geq \emptyset$ and therefore, the sum of dimensionalities of all subspaces discovered can be larger than the dimensionality of the data space.

We take a bottom-up approach to find the clusters in subspaces starting from finding base clusters in low dimensional subspaces. This approach integrates our proposed similarity measure into the algorithm. The base clusters indicate the similarities in low dimensional subspaces and their aggregation guarantees the similarity in higher dimensional spaces according to our Lemmas 4.3.2, 4.3.3, and 4.3.4.

One of the main aspects that differentiate SCBS and other bottom up algorithms such as CLIQUE, MAFIA and ENCLUS, is the idea of starting the search for similarity in $2D$ base subspaces instead of in individual dimensions. This method allows our algorithm to preserve the covariances between dimensions, which avoids the incorrect early dismissal of clusters, as discussed and illustrated in Section 4.5. Moreover, although the higher number of $2D$ base subspaces ($\frac{n(n-1)}{2}$ without subspace sampling) compared to the number of $1D$ subspaces (n) leads to more expensive computation in the first phase, it reduces the large number of clusters, which can raise challenges in interpreting the clustering results. Mullet et al. [160] show that CLIQUE and SUBCLU tend to produce a large number of clusters, which can also be observed in our experiments.

Another distinctive difference is in the aggregation of base clusters, which is done in a one-off frequent pattern mining task instead of as a sequential process. CLIQUE and SUBCLU expand the subspace by greedily adding one dimension at a time. Moreover, SUBCLU performs DBSCAN for each new dimension being added as the cluster subspace grows while SCBS performs clustering only once in a set of base subspaces. This reduces our time complexity and allows our algorithm to cluster datasets that SUBCLU cannot finish during the same running time. In addition, CLIQUE and SUBCLU use a static set of parameters for data density as the subspace dimensionality grows, which is identified as a limitation of these algorithms [128, 167] since the concept of distance and density change as the number of dimensions increases. In contrast, SCBS performs the clustering task only once in a set of base subspaces in the first phase, hence the parameters that are used for clustering are applied in subspaces that are equal dimensional.

Our lemmas support distance based, density based and grid based clustering. Therefore a wide range of different clustering algorithms [236] can be used for the first phase of the algorithm. We use kmeans as the underlying clustering algorithm for the evaluation due to its simplicity and the single additional parameter k that is required. In fact, an essential issue that needs to be properly addressed is the choice of parameters that the underlying clustering algorithm introduces and how they affect the overall algorithm. To this end, we also discuss in the evaluation the impact as well as the sensitivity of the parameter k in terms of the numbers and dimensionalities of the subspace clusters in the final results.

5.3 Proposed Method

We propose a two-phase subspace clustering algorithm as summarized in Algorithm 5.

Algorithm 5: Bottom-up clustering using FP-Trees.

Input : $X \in \mathbb{R}^{n \times d}$
p: dimensionality of base subspace
cluster_algo: the underlying clustering algorithm to use in base subspace
min_cluster_size

Output : $FP = \{F_i : i = 1..q\}$: set of frequent patterns that represent the clusters

`// Phase 1: search for base clusters in lower dimensional subspaces`

1 $all\_base\_subspaces = combination(d, p)$
2 $Z = 0_{n,|all\_base\_subspaces|}$
3 **foreach** S' *in all_base_subspaces* **do**
4 $projected\_X$ = project(X,S')
5 $cluster_{S'} = cluster\_algo(projected\_X)$
6 $Z_{*,S'} = cluster\_index_{S'}$
7 **end**

`// Phase 2: Extract clusters from FP-Tree`

8 $min\_sup = \frac{min\_cluster\_size}{n}$
9 T = build_fp_tree(Z,*min_sup*)

`// Analyze frequency at each level to prune T`

10 T = prune_tree(T)
11 $\{FP_i, Z_i\}$ = extract_maximal_frequent_sets(T)
12 *cluster_index* = check_membership(Z,FP)
13 *cluster_subspace* = refine_subspace(FP)
14 **return** *cluster_index,cluster_subspace*

5.3.1 Phase 1: Base Cluster Search

Our first phase searches for lower dimensional clusters in *base subspaces* (the concept is first defined in Chapter 4). These are called *base clusters* as they are the basis that form higher dimensional clusters. Unlike traditional bottom-up subspace clustering algorithms such as CLIQUE [14], ENCLUS and MAFIA [167] that search for dense units in individual dimensions, we search for base clusters in subspaces with two or more dimensions. A key advantage of searching for base clusters in subspaces instead of individual dimensions is that it enables us to consider covariances between dimensions. This is in accordance with our proposed method to measure similarity in high dimensional spaces.

Clustering is performed in each base subspace (lines 3-5 in Algorithm 5), and the cluster membership of each point in each base subspace is recorded in matrix Z (line 6). Table 5.2 shows a simplified example of the output Z of the first phase. In this example, the algorithm is applied on a dataset of five points $\{x_1, .., x_5\}$ in a 4-dimensional space. This illustrative example can correspond to observations collected by four sensors during five different times of the day, and the task is to discover if there is any group of observations that exhibit some common patterns at some locations, which is a problem of subspace clustering. Using $p = 2$,

the search for base clusters is performed in six base subspaces $\{d_1,d_2\}$, $\{d_1,d_3\}$,$\{d_1,d_4\}$, $\{d_2,d_3\}$, $\{d_2,d_4\}$, $\{d_3,d_4\}$, which are denoted as $\{S_1,..,S_6\}$ in the table. $C_{S_i,j}$ denotes the j^{th} cluster in the subspace S_i. Table 5.2 says that points x_1, x_2 and x_3 belong to the same cluster $C_{S_1,1}$ in subspace S_1. They also belong to cluster $C_{S_2,1}$ in subspace S_2, while sharing no common cluster in other subspaces.

Intuitively, the output Z provides the information of which points belong to which clusters in which base subspace, and can be used to study the proximity between points. However, each of its columns expresses only the proximity between points in a separate lower dimensional base subspace, and it is not straightforward to see which subset of points are similar and should be grouped to each other, and in which higher dimensional subspace. *The task of identifying the locally relevant subspace is significantly more challenging for clustering*, since we now cannot just consider one pair of points at a time as we showed in Chapter 4. Instead, the expected output is a set of subspaces, each of which is shared by a group of points. An indication of points belonging to the same cluster in high dimensional space is that they repeatedly belong to the same base clusters in base subspaces. Table 5.2 shows an example of points $\{x_1,x_2,x_3\}$ that can form a cluster in the higher dimensional subspace $\bigcup\{S_1,S_2,S_6\}$. We leverage this observation as well as our proven approach of aggregating base similarities (as presented in Lemmas 4.3.1-4.3.4) to propose an efficient way to combine base clusters to construct clusters in higher dimensional space in phase two.

5.3.2 Phase 2: High Dimensional Cluster Construction

Phase 2 learns the patterns of proximity among the points from the output Z of phase 1, to derive the final clusters and present them in a succinct and interpretable way. To this end, we consider Z as a transaction database where each point corresponds to a transaction and the base clusters covering that point are the items of that transaction. From Table 5.2, the first row is the transaction of point x_1, and the corresponding items are $C_{S_1,1},C_{S_2,1},C_{S_3,1},C_{S_6,1}$. The task of finding the groups of base subspaces that frequently cover a subset of points becomes the problem of mining the frequent patterns of items. Effectively, each frequent pattern

	Subspaces of base clusters					
Points	S_1	S_2	S_3	S_4	S_5	S_6
x_1	$C_{S_1,1}$	$C_{S_2,1}$	$C_{S_3,1}$	$\emptyset$	$\emptyset$	$C_{S_6,1}$
x_2	$C_{S_1,1}$	$C_{S_2,1}$	$C_{S_3,2}$	$C_{S_4,1}$	$\emptyset$	$C_{S_6,1}$
x_3	$C_{S_1}1$	$C_{S_2,1}$	$C_{S_3,3}$	$\emptyset$	$C_{S_5,1}$	$C_{S_6,1}$
x_4	$C_{S_1,2}$	$C_{S_2,2}$	$C_{S_3,4}$	$C_{S_4,2}$	$C_{S_5,1}$	$C_{S_6,1}$
x_5	$C_{S_1,2}$	$C_{S_2,2}$	$C_{S_3,4}$	$C_{S_4,2}$	$C_{S_5,2}$	$C_{S_6,1}$

Table 5.2 Base clusters in six subspaces of the dataset.

indicates a group of base clusters in different base subspaces that cover a *sufficiently large* group of points. Our Lemmas say that these points are similar and therefore form a cluster in the aggregated higher dimensional subspace. In Algorithm 5, we build a FP-Tree [97] from Z to mine the frequent patterns (line 9). Each branch of the tree is an aggregation of base clusters and represents a higher dimensional cluster. The sizes of the clusters are directly decided by the minimum support (`min_sup`) of the FP-Tree. For example, if `min_sup=0.1`, every cluster covers at least 10% of the points. In practice, it is more intuitive to know and decide the expected sizes of the clusters. Therefore, the algorithm takes this as a parameter, and lets it guide the value of `min_sup` (line 8). For example, when clustering hourly pedestrian events of one year's worth of data (8760 records), if we assume that any interesting pattern needs to be shared by at least 800 records to be a sufficiently strong signal to be considered, the expected minimum cluster size is 800 (i.e., any grouping of points that has less than 800 points is considered not stable enough). In this example, the minimum support is $\frac{800}{8760} = 0.091$.

Note that not all frequent patterns are useful as they can produce redundant clusters. For any cluster defined by the frequent pattern F_i, all subsets of F_i are also frequent [101], and correspond to clusters in lower dimensions, but none of them form a cluster as complete as F_i does. In this thesis, our focus is to find only clusters in the highest dimensional subspace that it can span. Therefore, we only need to mine the maximal frequent patterns (line 11) instead of finding all the frequent patterns. This significantly simplifies the algorithm and reduces the time complexity as presented in the next section.

In addition, it is important to control the number of frequent patterns since these can quickly grow. Prior to the extraction of maximal frequent patterns, phase 2 analyses the frequencies of patterns at different levels of the FP-Tree, and prunes small branches with low frequencies. These branches correspond to insignificant patterns and only reflect the characteristics of a small portion of the points that do not justify a cluster. This is essential to prevent the algorithm from producing a huge number of small clusters. To this end, phase 2 first performs a scan on the FP-Tree and records the frequency on each branch at each depth level of the tree. It then finds the knee-point [247], which indicates the level after which the frequencies significantly drop. Subsequently, the remainder of that branch is pruned.

After the maximal frequent patterns are mined, the remainder of the algorithm synthesizes the frequent patterns into interpretable clusters by defining the set of points and the subspace for each cluster. The point components of each cluster are identified using a simple membership check on line 12, i.e., each transaction of Z is checked if it contains any frequent pattern (each frequent pattern is equivalent to a cluster). Note that a transaction can match

to multiple patterns, especially short patterns. We give higher priority to clusters in higher dimensional subspaces, and hence prioritize to match longer frequent patterns first.

Next, the subspace in which each cluster resides is derived by aggregating the dimensions of all base subspaces (items) of the frequent patterns. In Chapter 4, each locally relevant subspace is refined by removing the duplicated dimensions (line 14 of Algorithm 2), the locally relevant subspace is applicable for a pair of points, hence any error is local to that pair only. In the case of clustering, a "wrong" instance of clustering in a base subspace can introduce new dimensions and increase the dimensionality of the subspace unexpectedly. This happens *only if the wrong base cluster covers a subset of transactions that happen to form a frequent pattern*. For example, if a sufficiently large set of points $\{x_i\}$ are grouped to the same cluster in base subspace (item) $\{d_1, d_2\}$, the item $\{d_1, d_2\}$ can incorrectly be part of a maximal frequent pattern in the FP Tree for this set of points; subsequently, dimensions d_1 and d_2 can potentially be included in the final higher dimensional subspace of the cluster. We empirically find that this is more likely to happen if large base clusters are found in the first phase, e.g., if a small value of k is used in k-means (k-means assigns every point to a cluster), or a large value of ε is used in DBSCAN. In addition to fine tuning the parameters of the first phase, it is also necessary to have a refinement strategy (line 13) to construct the subspace of the cluster. If a cluster is formed in a higher dimensional subspace S^*, its data points also form clusters in its lower dimensional base subspaces $S^{*'} \subset S^*$ according to the downward closure property. Therefore, the frequency of each dimension of S^* is expected to be sufficiently high. For example, if a set of points form a 3-dimensional subspace $\{d_1, d_2, d_3\}$, theoretically they also form clusters in $\{d_1, d_2\}$, $\{d_1, d_3\}$, $\{d_2, d_3\}$. Each dimension in this example has a frequency of 2. In general, the upper bound of the frequency is $|S^*| - 1$, where $|S^*|$ is the number of distinct dimensions of all base subspaces. In our algorithm, we set the cut off frequency threshold at $0.3 \times (|S^*| - 1)$.

We present a running example using Table 5.2 as the input, with `min_sup` set to 0.4. Figure 5.3 shows the FP-Tree before being pruned. The pruning eliminates the node C_{S_51}, which has a frequency of 1 (i.e., the patterns only apply to x_3) and hence should not justify a separate cluster. The branch that starts at node C_{S_51} on the right branch of the tree is also pruned (the patterns only apply to x_4). Eventually, two clusters are found as presented in Table 5.3.

5.3.3 Time Complexity Analysis

In the first phase, the search for base clusters in Algorithm 5 is performed in all p-dimensional base subspaces of the full dimensional space. Depending on the value of p, the number of base subspaces can be quadratic (when $p = 2$) or combinatorial to the dimensionality for

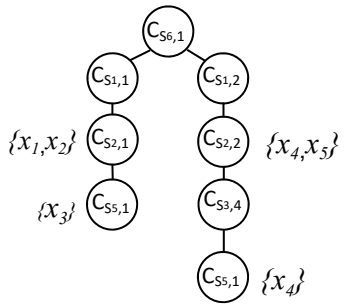


Fig. 5.3 FP-Tree built from Table 5.2

	Points	Patterns	Subspace
C_1	$\{x_1,x_2,x_3\}$	$\{C_{S_6,1},C_{S_1,1},C_{S_2,1}\}$	$\{d_1,d_3\}$
C_2	$\{x_4,x_5\}$	$\{C_{S_6,1},C_{S_1,2},C_{S_2,2},C_{S_3,4}\}$	$\{d_1,d_2,d_3,d_4\}$

Table 5.3 Clusters extracted from FP-Tree of the base clusters shown in Figure 5.3.

higher values of p. In each base subspace clustering, the computational complexity depends on the underlying clustering algorithm. k-means has time complexity of $O(n \times k \times p)$ (k is the number of clusters). Here, we are performing clustering in low dimensional base subspaces, hence p is small and can be considered constant; the time complexity of k-means hence becomes $O(n \times k)$. Similarly, the time complexity of DBSCAN is $O(n^2)$ in the worst case [193]. In order to keep the algorithm scalable to large volume inputs, we aim to keep the task of clustering base subspaces computationally inexpensive. The time complexities of popular clustering algorithms that can be applied in the first phase are described in the study of Xu et al. [236].

In phase 2, the process of building the tree and mining the maximal frequent patterns requires two traversals over the matrix Z. These traversals are linear in the number of transactions n of Z (it is also the volume of the original input). In each transaction, the algorithm needs to check all of its items. In the worst case, each point belongs to a base cluster, i.e., all items in every transaction are not null, and the algorithm needs to check all $\frac{d(d-1)}{2}$ base clusters for each row. Therefore, the upperbound of the time complexity of the second phase is $O(n \times d^2)$ (when $p = 2$).

It can be observed that the bottleneck of the algorithm is in its quadratic growth with respect to the dimensionality of the input. This affects both the search for base clusters in phase 1 and the mining of maximal frequent patterns in phase 2. One of the most straightforward methods to reduce this complexity is to reduce the number of base subspaces using a sampling approach. We have proven in Section 4.3 that it is not necessary to exhaustively check all base subspaces. Instead, as long as each dimension is checked at least α times, the proximities between points are still guaranteed. To this end, we perform sampling *with constraints on the frequencies of dimensions being sampled*, to ensure that

Algorithm 6: Phase 1 with sampling.

Input : $X \in \mathbb{R}^{n \times d}$
dim_min_freq: minimum number of appearance of a dimension
Output : $Z_{n,s}$

$all\_base\_subspaces = combination(d, p)$
shuffle$(all_base_subspaces)$
`// record the frequency of each dimension`
$dimension\_tally = O_{1 \times d}$
foreach S' *in* $all\_base\_subspaces$ **do**
  $dimension\_tally_{S'} += 1$
  $projected\_X$ = project(X,S')
  $cluster\_index_{S'}$ = cluster($projected\_X$)
  $Z_{*,S'} = cluster\_index_{S'}$
  `// check for early termination`
  if $min(dimension\_tally) > dim\_min\_freq$ **then**
    break
  end
end
return Z

all dimensions are sufficiently sampled. The first phase of the algorithm using the sampling approach is described in Algorithm 6

Using the sampling approach, the *minimum cluster size* and *minimum support* do not change since the number of points to be clustered remains the same. Sampling might reduce the robustness of the similarities in higher dimensional space, i.e., the similarity radius is not as tightly constrained (Lemma 4.3.2), or the probability that the points are close in higher dimensional space may reduce (Lemma 4.3.3). We show in our evaluation that sampling can significantly reduce the running time while still producing results comparable with the original algorithm.

5.4 Evaluation

We use a variety of synthetic datasets of different sizes and dimensions in order to evaluate our proposed algorithm's capability in finding clusters in disjoint and non-disjoint subspaces. The results are compared with traditional bottom-up clustering algorithms [160] as well as other state-of-the-art subspace clustering algorithms [76, 139]. We then evaluate our proposed algorithm that uses the subspace sampling technique ($SCBS_s$) in comparison to the original $SCBS_o$ algorithm, and discuss the effect and trade off of subspace sampling.

Subsequently, we evaluate the algorithm's scalability to inputs that have large volumes and high dimensionalities. All experiments are conducted with MATLAB on an Intel Core i7-4790 3.6GHz CPU and 16GB of RAM.

We use the following metrics to report the subspace clustering quality:

- NMI reports the quality of the grouping of points (row) in comparison with the cluster indices in the ground truth. NMI is commonly used to report clustering quality but has the limitation that it penalizes solutions less if they contain multiple small clusters [16, 186].
- F1 Score reports the cluster quality by measuring the effectiveness of the algorithm in grouping pairs of points that have the same ground truth index into the same cluster. F1 Score is able to address the limitation of NMI on results with large numbers of clusters.
- Subspace cluster Rand Index (RI) [168, 160] reports the cluster qualities by assessing both the segmentation of points to clusters, and the assignment of dimensions to the subspace of clusters. To this end, a point is considered to match the cluster ground truth if (1) it belongs to the correct cluster, and (2) its subspace matches the subspace in the ground truth; every mismatch of dimensions is penalized accordingly. Note that this metric is different from the more commonly known Adjusted Rand Index used for traditional low dimensional clustering [204]. Subspace cluster Rand Index can be used in conjunction of NMIs and F1 Scores to give a more holistic assessment. For example, if NMI is high but RI is low, it can be deduced that the grouping of points (rows) into clusters is good but the quality of the subspace (grouping of columns) is not good.
- The number of clusters and the average dimensionalities provide more insights on the returned clustering output.

5.4.1 Synthetic Data Generation

We generate synthetic datasets by first creating c spherical clusters such that the cluster centers are at equal distances and equal angles from the origin in the data space of dimension d_c. Geometrically, the centers are the vertices of a d_c-dimensional simplex [179, 96]. At each center, a normal distribution of n_c data points with standard deviation r is generated to form the corresponding cluster. At this stage, all clusters are formed in the same d_c-dimensional space. In order to emulate the challenge of local feature relevance, we introduce irrelevant dimensions and enforce that each cluster resides in a different subspace. Specifically, to generate clusters in non-disjoint subspaces, we add more dimensions (columns) to the dataset such that cluster

C_i resides in a subspace S_i constituted of dimensions $\{d_{i\times d_c}, d_{i\times d_c+1}, .., d_{(i+1)\times d_c-1}\}$. In this case, the synthetic dataset consists of $c \times n_c$ data points and $c \times d_c$ dimensions. In Section 5.4.4, we allow the subspaces to share some dimensions in order to evaluate the algorithm's ability to find clusters in non-disjoint subspaces. For most of the synthetic datasets, we generate equal size and equal dimensionality clusters. The sizes, dimensionalities, and the overlaps between the subspaces in which clusters reside are configurable and adjusted depending on the evaluation. We have full control on the properties of the synthetic datasets, including the number of clusters, density of the points and number of dimensions, and use this information to set the most appropriate parameters for all algorithms used in this evaluation.

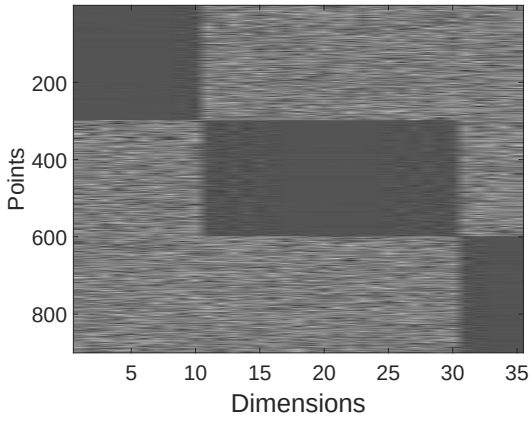
Figure 5.4a shows the grayscale heatmap of a sample dataset that has three clusters containing 900 points in a 35-dimensional space. The points $\{x_i\}_{i=1}^{300}$ follow a normal distribution with the mean μ_1 being the vertex of the simplex and standard deviation of 1. They form a cluster in the subspace $S_{1:10}$, which is constituted by the dimensions $\{d_1, ...d_{10}\}$. This set of points follow a uniform distribution in the range $[\mu - 5, \mu + 5]$ in the subspace $S_{11:35}$. Points $\{x_i\}_{i=301}^{600}$ form a cluster in the subspace $S_{11:30}$, which is constituted by the dimensions $\{d_{11}, ..., d_{30}\}$. Similarly, points $\{x_i\}_{i=601}^{900}$ form a cluster in the subspace $S_{31:35}$. The subspaces of these three clusters intersect only at the origin and hence are disjoint. On the other hand, Figure 5.4b is an example of data having clusters residing in non-disjoint subspaces, in which `Cluster 1` spans subspace S_{6789} and `Cluster 2` spans subspace S_{5678}.

The ground truth of the dataset comprises the indices of the cluster to which a point belongs. For example, the ground truth of the subset of points $\{x_i\}_{i=1}^{300}$ is 1, the ground truth of points $\{x_i\}_{i=301}^{600}$ is 2. This ground truth is directly used to compute the NMI and F1 Score. In addition, the ground truth also contains the dimensions of the subspaces, which is used to assess the average dimensionalities of clusters, and is used in conjunction with point indices to compute the subspace RI.

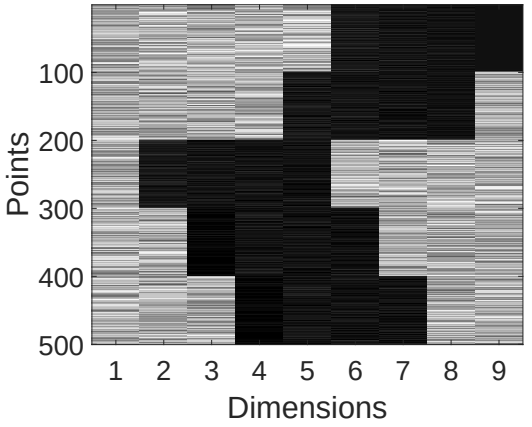
5.4.2 Evaluation of Clustering in Disjoint Subspaces

In this section we benchmark our algorithm with clustering algorithms CLIQUE, SUBCLU, DOC, and STATPC [160], as well as state-of-the-art algorithms SSC [76], LRR [139], and SSWC [50]. The implementation of CLIQUE, SUBCLU, DOC, P3C, and STATPC are redeveloped and published by Emmanuel et al. in the Weka OpenSubspace package[1] [159, 158]. The algorithms SSC, LRR, and SWCC are implemented in MATLAB and provided by the authors.

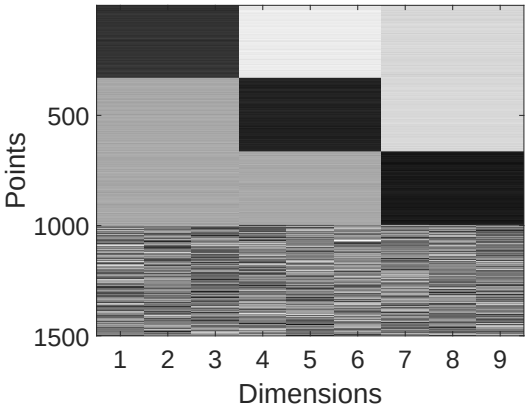
[1]The supplement material in this paper is no longer available. Its archive can be retrieved in https://archive.org/.



(a)



(b)



(c)

Fig. 5.4 (a) Clusters in disjoint subspaces with no outliers. Points $\{x_i\}_{i=1}^{300}$, $\{x_i\}_{i=301}^{600}$, $\{x_i\}_{i=601}^{900}$ form 3 clusters in 3 different subspaces, (b) Clusters in non-disjoint subspaces with no outliers, (c) Clusters in disjoint subspaces with 50% outliers.

The configurations of the synthetic dataset generation process are used to derive the most appropriate range of parameters for the algorithm, including the expected cluster numbers, sizes, average dimensionalities, and data density. For the algorithms that are implemented in Weka OpenSubspace, we replicate the hyperparameter grid search as described in [160]. More specifically, the parameters of CLIQUE, DOC, SUBCLU, and STATPC are as follows:

- The parameters ξ and τ of CLIQUE control the number of bins per dimension and the density threshold, respectively. In this evaluation, we also use the hyperparameter configuration of $x_i \in [5,30]$ and $\tau \in \{10^{-3}, 10^{-2}, 10^{-1}\}$.

- DOC has four important parameters: the parameter α controls the minimum number of points to form a cluster, β controls the balance between the number of points and number of dimensions in a cluster, k specifies the number of clusters, and w determines the maximum size of the cluster in a dimension (the width of the hypercube). We re-use the same search space of $\alpha \in [10^{-2}, 10^{-1}]$, that is, any group of points that has more than 10 points can form a cluster, and $\beta \in [0.1, 0.4]$. The parameter k is set to 5, and w varies in the range $[0.5, 1, 1.5, 2]$ to suit the distribution of the data points in the synthetic dataset.

- The *minPoints* parameter of SUBCLU that controls the number of points in a dense area is set to $minPoints \in [2^1, 2^2, 2^3, 2^4, 2^5]$. In addition, the search space for the parameter ε is $[0.1, 0.2, 0.4, 0.6, 0.8, 1]$

- STATPC requires three parameters of α_0, α_h, and α_k, which are the three significance levels used by the algorithm. Since it is not trivial to derive the direct relationship between these parameters and the distributions of our synthetic dataset, we reuse a scaled down version of the search configuration used in [160]: $\alpha_0, \alpha_h, \alpha_k \in \{10^{-12}, 10^{-8}, 10^{-4}, 1\}$. The search space excludes two very small values of $10^{-20}, 10^{-16}$ in the original paper in order to reduce the number of experiments to run.

The parametrization of three algorithms SSC, LRR, and SWCC is as follows:

- SSC is provided with the correct number of clusters in every experiment.

- In addition to the number of clusters, LRR uses a parameter λ to balance the weights of outliers and the legitimate input. In the first evaluations with synthetic data having no outliers, we use very low values of λ in the range of $[0, 0.05]$. As the outliers are introduced to the input in other experiments, the search space of λ is increased to $[0, 0.3]$ with the step of 0.01.

- In addition to the number of clusters and the number of subspaces, SWCC uses a parameter η to regularize its optimization problem of segmenting points and subspaces into clusters. This parameter has a direct impact on the weights of the output matrix that assigns a point and a dimension to a cluster. The weights concentrate on a few parameters if η is too small (lower number of smaller clusters), and are more evenly distributed if η is too large [50] (the clusters are less distinctive). The search space for this parameter is set to $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$ in this evaluation and should provide varying regularization settings for this experiment.

Our SCBS algorithm starts the search for base clusters in $2D$ subspaces ($p = 2$). k-means is used to find base clusters in phase 1. Two parameters are required for our algorithm, which are the number of base clusters k in each subspace, and the *minimum cluster sizes*. We set the expected minimum cluster sizes to 20% of the actual size of the cluster, e.g., if the ground truth is a dataset of 1000 points containing 5 clusters of 200 points each, any cluster found by SCBS that has more than 40 points satisfies this criterion. The minimum support used for the construction of FP-Tree in this example is $\frac{40}{1000} = 0.04$. Setting an appropriate value for k is non-trivial. As we presented earlier, the purpose of phase 1 is to find the similarity in cluster membership of the points in the low dimensional subspaces, rather than the exact cluster of each point. We invoke 12 iterations of our algorithm with $k \in \{3, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55\}$ and record the results.

We use the synthetic datasets as ground truth and evaluate the results on the cluster memberships of the points, as well as the dimensions of each subspace in which a cluster resides. In this experiment the size of the input is fixed to 1000 points, and the number of dimensions varies from 10 to 100. We acknowledge that these datasets have fairly small volumes and only medium dimensionalities. However, this setting allows more algorithms to finish within our running time limit, and allows our comparison to be more holistic. The running time limit is set to 30 minutes. Since the algorithms are implemented using different technologies and programming languages, we do not compare the time complexity in this experiment. Instead, this is evaluated and discussed in greater detail in the scalability analysis. The results are summarized in Table 5.4. Note that the analysis of the dimensionalities of subspaces, as well as the metric RI, are omitted for the algorithms SSC and LRR. The subspaces found by these algorithms are arbitrarily-oriented, i.e., each dimension of the discovered subspace is a linear combination of existing dimensions and it is not trivial to derive the subspace itself, its dimensionality, and its component dimensions. Moreover, the details on subspaces discovered by SSC and LRR are not discussed in the papers nor produced by the algorithm implementations.

It can be observed from Table 5.4 that our algorithm produces comparable or better results compared to SSC and SWCC across all the datasets, reflected in higher NMI, F1 Scores, and RI. The NMI indicates the homogeneity of the points in clusters, F1 Scores take into account and balance all the correct and incorrect groupings, while RI reflects both the quality of clustering points and the discovery of subspaces. All these three metrics consistently indicate that our algorithm performs better in the experiments with $d \in \{10, 40, 50, 70, 80\}$. Table 5.4 also shows that although SCBS achieves better RI values, SWCC tends to have more stable values of this metric, reflected in both higher overall minimum RI values, and smaller gaps between minimum and maximum RI values. We believe the reason is that SWCC is provided with the correct numbers of clusters and numbers of subspaces. We observe that RI values tend to decrease as the number of clusters deviates from the ground truth. Here, every additional cluster increases the chance of subsets of dimensions not being grouped to the same subspace, and subsets of points not being grouped to the same cluster. Subspace RI therefore penalizes the clustering result more. STATPC can produce clusters that have comparable NMIs and F1 Scores in some experiments but the results have significantly lower RI scores, which can be explained by the large number of clusters that STATPC usually produces. This also explains the very low values of RI of CLIQUE, despite the moderate NMIs and F1 Scores.

SCBS, SSC, SWCC and STATPC are the only algorithms that can run to completion within the time threshold. DOC gives consistently high accuracy provided that all five parameters of the algorithm are well-tuned. However, it has significant higher running time and cannot cluster data larger than 1000×40 within 30 minutes.

The numbers of clusters produced by SCBS are fairly stable and vary in the range $[2, 9]$, in comparison to the ground truth of 5 clusters. Through the experiments, we found that the low numbers of subspace clusters are usually associated with low values of k in the first phase, e.g., $k = 3$. This choice of k parameter results in small numbers of large groupings of points in the base subspaces, i.e., each item (base cluster) has high frequency (covers multiple points), which subsequently leads to low numbers of long frequent patterns being mined in the second phase. These frequent patterns also usually contain multiple items, corresponding to the clusters that reside in higher dimensional subspaces. In addition to this, we also further analyze the effects of k on the clustering quality in this experiment, as shown in Figure 5.5. It can be observed that as k increases the values of NMI also increase and peak at $k = 30$, after which the results deteriorate. Nevertheless, there is a wide range $[20, 45]$ of k in which the clustering results are reasonably stable (except for $k = 25$).

In contrast, high values of k lead to the input being split into multiple $2D$ clusters in the base subspaces, resulting in each base cluster covering fewer points, i.e., each item of the

transactions has lower frequency. To this end, the output of the second phase contains *a large number of shorter frequent patterns*, corresponding to a higher number of lower dimensional clusters. These impacts on the number and the dimensionality of subspace clusters are also applicable to any other parameters of other algorithms that can control the size of the base clusters in the first phase. For example, large values of ε and low values of *minPoints* of DBSCAN can result in large base clusters and lead to the same effect in the final clustering result.

The average dimensionalities of the clusters are also close to the ground truth. Using our mechanism of synthetic data generation, the ground truth dimensionality of clusters in this experiment is $d/5$ (five equal dimensional clusters are generated). Table 5.4 shows that this value falls in between the minimal and maximal average dimensionality values in every experiment. Note that our algorithm can find clusters in overlapping subspaces, hence the sum of all cluster dimensionalities can be higher than the number of dimensions of the data space. For example, in the experiment with $d = 100$, the maximal number of clusters found is 8, corresponding to the average dimensionality of 17.1 (the sum of the number of dimensions of all subspaces is 137 because some dimensions are counted more than once).

		d=10		d=20		d=30		d=40		d=50		d=60		d=70		d=80		d=90		d=100	
		min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max
SCBS	NMI	0.35	0.86	0.42	0.88	0.50	0.84	0.32	0.78	0.43	0.81	0.47	0.80	0.45	0.79	0.34	0.77	0.25	0.79	0.29	0.75
	F1 Score	0.51	0.92	0.57	0.93	0.64	0.91	0.48	0.86	0.58	0.88	0.61	0.88	0.59	0.88	0.49	0.86	0.42	0.88	0.45	0.84
	RI	0.05	0.64	0.01	0.84	0.03	0.74	0.04	0.76	0.04	0.79	0.06	0.79	0.11	0.72	0.05	0.79	0.03	0.79	0.04	0.75
	NumClusters	2	10	2	9	2	7	2	7	2	9	2	6	2	9	2	7	2	7	2	8
	Avg ClusterDim	2.2	5.0	3.0	4.0	3.3	5.5	6.7	9.3	13.0	15.0	13.2	14.0	16.4	17.0	15.1	20.0	16.1	20.0	17.1	23.0
SSC	NMI	0.42	0.82	0.55	0.79	0.35	0.73	0.32	0.67	0.38	0.73	0.44	0.87	0.24	0.79	0.37	0.73	0.25	0.83	0.26	0.73
	F1 Score	0.57	0.89	0.68	0.87	0.50	0.82	0.47	0.77	0.53	0.83	0.59	0.93	0.41	0.87	0.53	0.83	0.41	0.90	0.43	0.83
	RI	not applicable																			
	NumClusters	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	Avg ClusterDim	not applicable																			
SWCC	NMI	0.43	0.79	0.22	0.90	0.34	0.84	0.30	0.75	0.20	0.73	0.50	0.74	0.56	0.61	0.48	0.76	0.18	0.52	0.40	0.81
	F1 Score	0.58	0.87	0.39	0.94	0.50	0.91	0.47	0.84	0.37	0.83	0.64	0.83	0.68	0.73	0.62	0.84	0.35	0.65	0.55	0.88
	RI	0.32	0.69	0.18	0.69	0.34	0.51	0.33	0.53	0.33	0.58	0.48	0.55	0.53	0.57	0.48	0.61	0.17	0.54	0.26	0.76
	NumClusters	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
	Avg ClusterDim	1.8	3.0	3.8	4.75	5.8	7.2	7.8	9.5	9.8	9.8	11.8	13.4	12.2	15.0	15.2	17.4	16.8	19.2	17.2	19.8
LRR	NMI	0.12	0.79	0.38	0.87	0.53	0.74	0.55	0.60	0.17	0.25	0.12	0.16	0.29	0.62	-	-	-	-	-	-
	F1 Score	0.30	0.87	0.53	0.92	0.67	0.83	0.68	0.72	0.34	0.42	0.30	0.34	0.45	0.74	-	-	-	-	-	-
	RI	not applicable																			
	NumClusters	5	5	5	5	5	5	5	5	5	5	5	5	5	5	-	-	-	-	-	-
	Avg ClusterDim	not applicable																			
STATPC	NMI	0.22	0.80	0.49	0.70	0.31	0.71	0.14	0.68	0.37	0.73	0.18	0.71	0.23	0.77	0.09	0.73	0.15	0.67	0.27	0.70
	F1 Score	0.39	0.88	0.63	0.80	0.47	0.81	0.32	0.78	0.52	0.82	0.35	0.81	0.40	0.86	0.28	0.83	0.33	0.78	0.44	0.81
	RI	0.23	0.77	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.10	0.01	0.01	0.01	0.01	0.01	0.02
	NumClusters	5	43	12	25	13	31	9	57	15	29	12	47	7	41	10	122	12	63	9	37
	Avg ClusterDim	1.3	10	1.2	20	2.7	30	3.5	40	7.3	50	8.0	60	8.3	70	8.3	80	8.7	90	11.9	100
CLIQUE	NMI	0.10	0.72	0.06	0.47	0.12	0.41	0.10	0.25	0.02	0.29	0.03	0.21	0.02	0.31	0.01	0.10	0.01	0.15	0.02	0.19
	F1 Score	0.28	0.82	0.25	0.61	0.30	0.56	0.29	0.42	0.22	0.45	0.23	0.38	0.22	0.47	0.21	0.28	0.21	0.33	0.22	0.36
	RI	0.01	0.03	0.02	0.03	0.01	0.10	0.01	0.01	0.02	0.09	0.01	0.01	0.01	0.01	0.01	0.01	0.04	0.08	0.01	0.01
	NumClusters	3	108	7	403	19	97	77	120	65	726	34	519	41	604	25	811	117	1003	92	602
	Avg ClusterDim	1.1	1.8	1.7	2.2	2.3	3.5	2.4	3.9	1.1	5.2	1.9	4.1	2.5	8.2	2.3	8.1	1.9	11.4	2.6	15.5
DOC	NMI	0.67	0.82	0.61	0.85	0.58	0.87	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	F1 Score	0.78	0.89	0.73	0.92	0.70	0.92	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	RI	0.01	0.02	0.01	0.02	0.01	0.03	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	NumClusters	5	5	5	5	5	5	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Avg ClusterDim	1.9	2.3	2.1	2.7	3.3	4.1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SUBCLU	NMI	0.41	0.64	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	F1 Score	0.56	0.75	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	RI	0.01	0.12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	NumClusters	7	303	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Avg ClusterDim	1.9	5.1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 5.4 Evaluation of algorithms on synthetic datasets. The volume is fixed to 1000 data points and its dimensionality varies in the range $[10, 100]$. The best result for each dataset is highlighted.

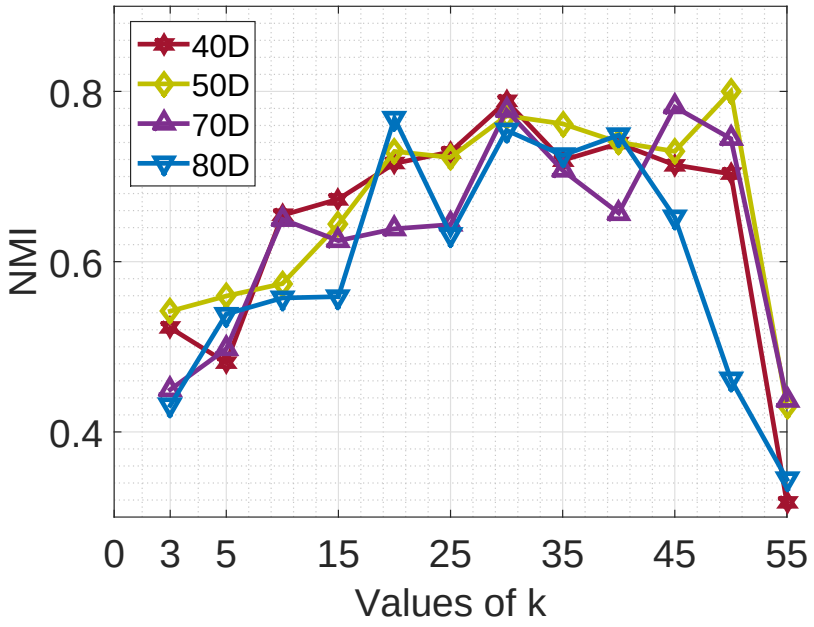


Fig. 5.5 Evaluation of the algorithm sensitivity to the parameter k on synthetic datasets.

The previous evaluation is done on data with no outliers, and every point belongs to a cluster in the ground truth. We also compare the algorithms on their ability to work with data having different levels of outliers. Outliers are points that are distant from most other observations and follow random distributions [163]. In practice, outliers can be introduced to the data for various reasons, such as legitimate anomalous events, sensor malfunction, or transmission error [163]. We add outliers to the synthetic datasets by generating a uniform distribution in a range that is different from that of the clustered points. To this end, we use the dataset having 50 dimensions generated in the previous evaluation as the starting point, and introduce outliers to it for this experiment. The levels of outliers vary in the range of $[0\%, 60\%]$. Here, 50% of outliers correspond to 500 points that follow a uniform distribution in the range $[\mu - 5, \mu + 5]$, where μ is the centre of the clusters. The final datasets used for evaluation have a volume in the range of $[1000, 1500]$ points. Figure 5.4c shows the grayscale heatmap of an example of a dataset of 1000 points that belong to some clusters, mixed with 500 points of outliers. In this experiment, the cluster numbers of SSC, LRR, and SWCC are set in the range $[5, 10]$ to take into account the possibility that these algorithms might be able to correctly group the outliers into their own clusters.

The results are shown in Table 5.5. In the implementation of this experiment, the outliers are labelled as "cluster index" 0 in the ground truth, and are excluded from the computations of NMI, RI and F1 Scores. It is arguably easier not to assign these outliers to any clusters,

i.e., the algorithm correctly sets their indices to 0, hence including them gives an unrealistic increase to the metrics. Note that if a point x_i (where $i < 1000$) is incorrectly identified as an outlier, the wrong clustering *is still penalized*. Table 5.5 shows that outliers at low levels have little effect on the clustering results of our proposed algorithm. As the outlier levels are less than 40%, the metrics only slightly drop (except for $outlierLevel = 20\%$), and the numbers as well as dimensionalities of the clusters remain the same. Further inspection of the results shows that an outlier is excluded from being clustered in two phases:

- An outlier does not belong to any cluster in the base subspace and hence naturally does not belong to any subspace cluster.
- If an outlier is still grouped to a cluster in a certain base subspace by chance in the first phase, it is unlikely that this consistently occurs in all the base subspaces that are part of a frequent pattern. Thus the transaction corresponding to an outlier does not match a frequent pattern, i.e., the outlier does not belong to a high dimensional subspace cluster.

As the levels of outliers increase past 50%, the metrics, although they are still higher than those of the other algorithms, start to drop noticeably. We observe that at high levels of outliers, the outliers start to form their own dense areas and can be misclassified as clusters in base subspaces. This happens frequently enough that these base clusters form their own frequent pattern and become a higher dimensional subspace cluster. At high levels of outliers, this becomes more likely with small values of k in k-means, which assigns every point to a base cluster.

SSC and LRR express each point as a linear combination of other points and subsequently solve optimization problems to achieve the segmentation of points into clusters. For this reason, each outlier can still play a role in the optimization equation and impact the clustering results. A more detailed analysis on the impact of high levels of outliers to SSC is available in [202]. Although LRR has the parameter λ to balance the weights between outliers and legitimate cluster points (and we vary λ in the range of $[0, 0.3]$ in each of these experiments), this does not show a significant impact on the results.

5.4.3 Evaluation of SCBS with Sampling Approach

We evaluate the effectiveness of our clustering algorithm with the subspace sampling approach ($SCBS_s$, with the subspace sampling method described in Algorithm 6). We apply $SCBS_s$ on the datasets generated for the experiments in Section 5.4.2 and compare the results with the results generated by the original algorithm ($SCBS_o$ as described in Algorithm 5).

		Levels of outliers													
		0%		10%		20%		30%		40%		50%		60%	
		min	max	min	max	min	max	min	max	min	max	min	max	min	max
SCBS	NMI	0.43	0.81	0.41	0.81	0.41	0.76	0.35	0.78	0.34	0.77	0.30	0.71	0.28	0.63
	F1 Score	0.58	0.88	0.56	0.88	0.56	0.85	0.50	0.87	0.50	0.86	0.46	0.81	0.44	0.75
	RI	0.04	0.79	0.00	0.81	0.01	0.76	0.01	0.74	0.01	0.75	0.03	0.82	0.01	0.67
	NumClusters	2	10	2	10	2	10	2	10	2	10	3	10	3	10
	Avg ClusterDim	13.0	15.0	13.0	15.0	13.0	15.0	13.0	15.0	13.0	15.0	11.6	13.0	13.0	13.3
SSC	NMI	0.27	0.73	0.24	0.66	0.21	0.58	0.17	0.45	0.16	0.47	0.14	0.43	0.15	0.40
	F1 Score	0.40	0.83	0.41	0.77	0.38	0.71	0.35	0.60	0.33	0.61	0.32	0.58	0.33	0.55
	RI	not applicable													
	NumClusters	5	10	5	10	5	10	5	10	5	10	5	10	5	10
	Avg ClusterDim	not applicable													
SWCC	NMI	0.16	0.81	0.15	0.77	0.14	0.72	0.14	0.69	0.11	0.62	0.11	0.61	0.10	0.59
	F1 Score	0.30	0.89	0.33	0.86	0.32	0.82	0.32	0.79	0.30	0.74	0.29	0.73	0.29	0.71
	RI	0.02	0.73	0.01	0.89	0.00	0.83	0.01	0.82	0.01	0.76	0.01	0.74	0.02	0.73
	NumClusters	5	10	5	10	5	10	5	10	5	10	5	10	5	10
	Avg ClusterDim	4.5	9.8	4.5	9.8	4.7	10.2	4.3	9.5	4.1	9.2	3.7	9.2	3.7	9.2
LRR	NMI	0.16	0.25	0.15	0.23	0.11	0.19	0.11	0.16	0.07	0.15	0.09	0.15	0.09	0.12
	F1 Score	0.29	0.42	0.33	0.40	0.29	0.36	0.29	0.34	0.26	0.33	0.28	0.33	0.28	0.30
	RI	not applicable													
	NumClusters	5	10	5	10	5	10	5	10	5	10	5	10	5	10
	Avg ClusterDim	not applicable													

Table 5.5 Evaluation of algorithms on synthetic datasets that have 1000 data points, 50 dimensions and different levels of outliers. The best result for each dataset is highlighted.

The *frequency threshold* is set to $\sqrt{d}$ so that the number of base subspaces to be sampled increases more slowly than the quadratic growth of the dimensionality d. More specifically, the expected time complexity in regard to the dimensionality in this case is $O(d \times \sqrt{d})$. The results are summarized in Table 5.6.

Table 5.6 shows that the metrics that purely measure point segmentation (NMI and F1 Score) slightly decrease compared to those produced by the original algorithm. However, the values of RI observe bigger reductions. In addition, the results also show a larger range of cluster numbers and cluster dimensionalities. This indicates that $SCBS_s$ is less accurate in determining the correct subspaces of clusters. We identify that the sampling approach can either wrongly include or wrongly exclude dimensions for a subspace in the following ways:

- First, it can cause clusters to tend to have higher dimensions, which subsequently increases the maximal average dimensionalities as shown in Table 5.6. The frequency threshold (i.e., dim_min_freq in Algorithm 6) sets a lower bound of the frequency of which each dimension needs to be sampled, however, there is no constraint on the upperbound. Therefore, some dimensions can be over-sampled as long as the frequencies of all other dimensions do not satisfy the cut off threshold yet. For example, with a threshold of 2, the following sequence subspace sampling $\{d_1, d_2\}$, $\{d_1, d_4\}$, $\{d_1, d_3\}$, $\{d_3, d_4\}$, $\{d_2, d_3\}$ only stops when all dimensions are sampled twice. In this

case, d_1 has been over sampled (its frequency is 3). Although the randomized sampling of subspaces (line 2 of Algorithm 6) can limit this effect, it is still inevitable with the current simple sampling technique. Subsequently, these oversampled dimensions are given more advantages in the refinement of subspaces (line 13 of Algorithm 5). One solution to this problem is to enforce more rules to the sampling technique, e.g., to enforce an upper bound on the frequencies of dimensions, or discard the sampled base subspace and re-sample if both of its dimensions already satisfy the dim_min_freq threshold. This extra logic not only can ensure the dimensions are more evenly sampled, but can also reduce the number of base subspaces being sampled, therefore further increase the algorithm's efficiency.

- Second, the sampling approach can also potentially reduce the dimensionalities of subspace clusters. It is possible that a dimension is not sampled together with a relevant dimension that it can be grouped with in any of the base clusters. For example, in the 1000×50 dataset, the sampling guarantees that each dimension is sampled at least 8 times; however, there is a chance that the dimension d_1 is never sampled with any of the dimensions $\{d_2, d_3, ..., d_{10}\}$. Therefore, although a cluster is found for the subset of points $\{x_i\}_{i=1}^{200}$, the dimension d_1 is not included in the final cluster subspace. With our configurations of the data generation, the number of clusters subspaces is $\frac{d}{5}$ (the number of clusters is 5) while the minimum frequency of a dimension being sampled is $\sqrt{d}$, which increases at a slower rate. Therefore, there is an increasing probability that *a dimension is never sampled with other relevant dimensions with which it can form a subspace*. This explains the low values of the minimum cluster dimensionality as d increases. This observation suggests a limitation of $SCBS_s$ with data that have *clusters with high dimensionalities*. For such datasets, the minimum sampling frequency of dimensions needs to be increased, which increases the algorithm's running times.

In this evaluation, the biggest drop of NMI of the clustering result is with $d = 100$, in which the best NMI decreases from 0.75 to 0.56. This translates to a 25% drop in cluster quality in return of a 64% decrease in computational complexity: 1782 base subspaces being inspected compared to 4950 subspaces in the original approach. The minimum number of base subspaces to be sampled theoretically can be lower than 1782, however some dimensions are over sampled due to our simple sampling approach as described above.

5.4.4 Evaluation of Clustering in Non Disjoint Subspaces

We next evaluate the capability of our algorithm to find clusters in non-disjoint subspaces. In this evaluation we use 1000 data points, where the number of dimensions varies between 10

		d=10		d=20		d=30		d=40		d=50		d=60		d=70		d=80		d=90		d=100	
		min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max
$SCBS_o$	NMI	0.35	0.86	0.42	0.88	0.50	0.84	0.32	0.78	0.43	0.81	0.47	0.80	0.45	0.79	0.34	0.77	0.25	0.79	0.29	0.75
	RI	0.05	0.64	0.01	0.84	0.03	0.74	0.04	0.76	0.04	0.79	0.06	0.79	0.11	0.72	0.05	0.79	0.03	0.79	0.04	0.75
	F1 Score	0.51	0.92	0.57	0.93	0.64	0.91	0.48	0.86	0.58	0.88	0.61	0.88	0.59	0.88	0.49	0.86	0.42	0.88	0.45	0.84
	NumClusters	2	10	2	9	2	7	2	7	2	9	2	6	2	9	2	7	2	7	2	8
	Avg ClusterDim	2.2	5.0	3.0	4.0	3.3	5.5	6.7	9.3	13.0	15.0	13.2	14.0	16.4	17.0	15.1	20.0	16.1	20.0	17.1	23.0
$SCBS_s$	NMI	0.32	0.85	0.39	0.79	0.43	0.78	0.27	0.70	0.35	0.70	0.36	0.68	0.36	0.70	0.24	0.64	0.22	0.62	0.22	0.56
	RI	0.01	0.35	0.00	0.58	0.02	0.68	0.02	0.61	0.01	0.61	0.01	0.67	0.00	0.62	0.01	0.59	0.01	0.68	0.01	0.50
	F1 Score	0.47	0.92	0.54	0.87	0.58	0.86	0.44	0.81	0.51	0.80	0.51	0.78	0.52	0.80	0.41	0.75	0.39	0.73	0.39	0.69
	NumClusters	2	10	1	10	2	11	2	13	1	12	1	12	1	12	1	12	1	13	1	15
	Avg ClusterDim	2.1	6.0	2.1	6	2.5	7.5	3.0	12.1	3.5	17	3.9	19.8	5.2	21.4	3.7	25.0	5.7	25.0	6.2	27.0

Table 5.6 Evaluation of the sampling approach ($SCBS_s$) in comparison to the original algorithm ($SCBS_o$). Note that $SCBS_o$ outperforms $SCBS_s$ in every experiment.

and 100, and the clusters reside in overlapping subspaces. Figure 5.4b shows an example of data having clusters in non disjoint subspaces. In this experiment, the overlapped subspaces between clusters are generated randomly. Specifically, we start the data generation with a $(c \times n_c) \times d_c$ matrix, corresponding to c clusters as described in Section 5.4.1, then randomly add columns to the left and right of each $n_c \times d_c$ block of the matrix, with the only constraint that the total number of columns is d. Eventually, the final dataset is a $(c \times n_c) \times d$ matrix, in which clusters reside in overlapping subspaces. The other algorithms that produce comparable results in the previous section are not included since they are not able to find clusters in overlapping subspaces: SSC and LRR are only designed to find clusters in disjoint subspaces [76][139]. Moreover, SWCC assigns weights for each column according to the column's membership to all clusters, and the weights of each column are summed up to 1. This indicates that the memberships of each column to different clusters are exclusive.

		d=10		d=20		d=30		d=40		d=50		d=60		d=70		d=80		d=90		d=100	
		min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max	min	max
SCBS	NMI	0.32	0.78	0.45	0.77	0.39	0.60	0.31	0.77	0.41	0.52	0.37	0.57	0.37	0.62	0.29	0.63	0.37	0.59	0.28	0.50
	RI	0.35	0.64	0.49	0.66	0.42	0.45	0.34	0.54	0.45	0.58	0.41	0.58	0.40	0.60	0.31	0.71	0.40	0.69	0.30	0.59
	F1 Score	0.48	0.86	0.59	0.85	0.54	0.72	0.47	0.85	0.56	0.66	0.53	0.69	0.52	0.74	0.45	0.75	0.52	0.71	0.44	0.64
	NumClusters	3	10	3	11	5	11	6	10	6	12	5	11	8	13	8	14	10	15	11	15
	Avg ClusterDim	2.0	4.3	2.6	4.3	3.2	5.2	5.0	6.3	5.8	9.5	7.5	12.2	7.0	9.1	7.7	11.9	7.4	10.1	8.3	10.9

Table 5.7 Evaluation of the algorithm in finding clusters in non-disjoint subspaces.

Table 5.7 shows that the clustering quality decreases compared to the clustering of data in disjoint subspaces presented in Section 5.4.2. Another common pattern observed is that the numbers of clusters increase while the average dimensionalities of clusters decrease from the previous results. This phenomenon can be explained by the overlapping subspaces as follows. The clusters found in the base subspaces can cover a sufficiently large subset of points (i.e., sufficiently frequent) to form frequent patterns that correspond to *only the overlapping subspace*; and these frequent patterns cover points that should have belonged to different clusters according to the ground truth. For example, the base clusters found in subspace $\{d_3, d_4, d_5\}$ of the dataset in Figure 5.4c can potentially form a frequent pattern that covers points $\{x_i\}_{i=200}^{400}$ since they cover 40% of the points. Such a cluster discovered by the algorithm is valid but is still penalized by our ground truth nevertheless: the ground truth

records that points $\{x_i\}_{i=200}^{300}$ and $\{x_i\}_{i=300}^{400}$ belong to two different clusters. The assignment of points to clusters is done by performing a membership check between the items of the points (transactions) and the items of the frequent patterns, with higher priority given to higher dimensional subspaces (line 12 of the Algorithm 5 - more details in Section 5.3.2). Hence, the above incorrect clustering only happens if the frequent patterns corresponding to overlapping subspaces contain more individual dimensions than other discovered frequent patterns. Using Figure 5.4c as an example, let S_1 denote the subspace d_2, d_3, d_4, d_5, and S_2 denote the subspace d_3, d_4, d_5. Here, S_1 is the ground truth subspace of the cluster C_1 that cover points $\{x_i\}_{i=200}^{300}$ while S_2 is the non-disjoint (overlapping) subspace of cluster C_1 and the cluster that cover points $\{x_i\}_{i=300}^{400}$. If the algorithm misses one or more dimensions of the subspace in which C_1 resides, this subspace has fewer dimensions and is given lower priority than subspace S_2, which subsequently leads to points $\{x_i\}_{i=200}^{300}$ being assigned to the overlapping subspace S_2. If no frequent pattern that corresponds to subspace $\{d_3, d_4, d_5, d_6\}$ is discovered, the points $\{x_i\}_{i=300}^{400}$ are also grouped into the same cluster C_1 in subspace $\{d_3, d_4, d_5\}$, i.e., points $\{x_i\}_{i=200}^{400}$ belong to the same cluster. While we believe it is acceptable for these points to be grouped to the same cluster in $\{d_3, d_4, d_5\}$, the ground truth is different and penalizes such a cluster solution.

In summary, the drops in NMI, RI and F1 Scores can be partially explained by the new clusters in the overlapping subspaces that are not recorded in the ground truth. These clusters are still valid; therefore the lower values of metrics should not be considered entirely as a decrease in clustering quality. However, composing the ground truth data that covers all the possible clustering of points is non trivial. For example, such a ground truth should record that points x_{250} and x_{350} should belong to *two different clusters*, but *it is still acceptable for them to be in the same cluster* in some situations. To this end, this can be regarded as two different sets of ground truths, and the total number of sets of ground truth required to fully evaluate all possible groupings of points grow combinatorial with the points. In this evaluation, after gaining insights on the reasons why the metrics decrease, we regard the high (albeit decreasing) values of NMI, RI and F1 Score as an indication that the cluster structures are still discovered by the algorithm, and that the algorithm is capable of finding clusters in non-disjoint subspaces.

5.4.5 Evaluation of Scalability

Scalability to the input volume

We evaluate the scalability of *SCBS* to the number of data points by generating data having 10 dimensions and varying the number of data points from 1,000 to 1,000,000. We include

Table 5.8 Scalability with the number of data points: the clustering algorithm is applied on datasets that have 10 dimensions and the number of data points varies between 10,000 and 1,000,000. The running times are recorded in minutes.

	n=5K	d=10K	d=15K	d=50K	d=100K	d=150K	d=300K	d=500K	n=1000K
SCBS	0.11	0.22	0.34	2.13	3.07	5.97	20.76	52.61	280.27
SWCC	1.47	3.02	4.72	15.33	30.83	46.92	91.24	148.87	353.60
SSC	3.77	6.95	25.11	-	-	-	-	-	-

only LRR, SSC and SSWC in this scalability evaluation because they are the fastest baseline algorithms with high accuracy. The number of dimensions is static so that the scalability of the algorithm can be evaluated with respect to the number of data points. While the numbers are kept to low values of 10, it helps shorten the total running times of this experiment, while still allowing us to differentiate the scalability of different algorithms. We also evaluate the scalability to the number of dimensions in the next section.

The execution times are presented in Table 5.8 and Figure 5.6. It shows that the running of SCBS to cluster 500,000 points is 52.6 minutes ($10^{3.5}$ seconds), in comparison with the running time of 148.9 minutes of SWCC. In addition, only SCBS and SWCC can run to completion with 1 million data points. SSC triggers memory errors when the number of points exceeds 15,000. The running time and required memory of LRR are even higher, and we cannot run LRR to completion to cluster 5,000 data points. Wang et al. [227] shows that the time complexity of SSC is at least quadratic to the input volume, while the time complexity of LRR to solve the optimization problem is $O(n^3)$ to the input volume. Our time complexity analysis shows that the time complexity of *SCBS* is linear to the number of data points, which can make it more scalable to very large datasets.

Scalability to the dimensionality

We evaluate the scalability of our proposed algorithm to the dimensionality of the input in this section. Since the original algorithm as presented in Algorithm 5 does not scale well with the dimensionality, we use the algorithm with the sampling approach ($SCBS_s$ - phase 1 is described in Algorithm 6) in this evaluation. The effectiveness of this algorithm is discussed in comparison to the original algorithm in Section 5.4.3.

In order to avoid the dataset having disproportionate volume and dimensionality, i.e., the volume is not large enough and makes the data more sparse as the number of dimensions increases, we use a reasonably large volume synthetic dataset with 10,000 data points and vary the number of dimensions d in range $[100, 3500]$. Although such large volumes are not necessary when d is low, it is important that the input volume is fixed and the scalability is

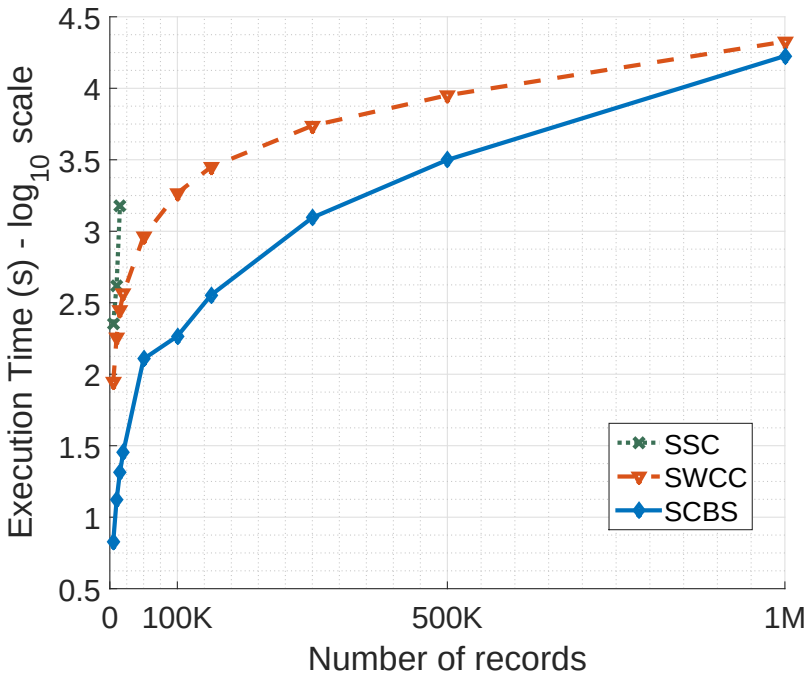


Fig. 5.6 Scalability with number of records. The time is in log_{10} scale.

evaluated against the changes in dimensionality in isolation. In summary, the details of the synthetic datasets are as follows:

- Each dataset has 10,000 data points and its dimensionality varies in range $[100, 3500]$.
- The subspaces are disjoint and have the same number of dimensions, the number of clusters is proportional to the total dimensionality such that each cluster resides in a 20-dimensional subspace, i.e., the 100-dimensional dataset has five 20-dimensional clusters.
- The clusters are generated using the same mechanism described in Section 5.4.1: the centres are first generated as vertices μ of a high dimensional simplex, and the clusters are then generated as a normal distribution $N(\mu, 1)$ in the subspace. The data in the irrelevant dimensions are generated as a uniform distribution in the range $[\mu - 5, \mu + 5]$.

We set the minimum threshold that each dimension needs to be sampled to be $\sqrt{d}$. Reflecting on the phenomenon that the dimension can be over sampled, which might increase the running time as discussed in Section 5.4.3, we apply an additional simple strategy to avoid this. Specifically, during the sampling process, if both dimensions of a sampled base subspace already have their frequencies satisfy the threshold, the base subspace is skipped

and the sampling continues. Due to the long running time, the algorithm is executed only once for each dataset, with the parameter k set to 30. This is the value of k that gives the highest NMI in most of the evaluation with synthetic datasets we have done so far, according to Figure 5.5. The running times and the numbers of sampled subspaces are recorded and summarized in Table 5.9. In this experiment, the mean and standard deviation of the NMI values achieved by the algorithm are 0.56 and 0.12, respectively (note that the algorithm is applied on each dataset only once with no grid search for optimal parameters).

		d=100	d=500	d=1000	d=1500	d=2000	d=2500	d=3000	d=3500
SCBS	Running time (m)	2.89	35.2	103.6	165.0	235.6	364.4	544.3	608.1
	Number of sampled base subspaces	636	6405	17,592	31,859	48,688	68,693	87,488	113,204
SWCC	Running time (m)	33.44	164.01	373.52	464.01	645.00	946.71	1219.67	1394.25
SSC	Running time (m)	39.37	-	-	-	-	-	-	-

Table 5.9 Scalability with the number of dimensions: the clustering algorithm is applied on datasets that have 10,000 data points and number of dimensions in range $[100, 3500]$.

Figure 5.7 shows that SCBS (with subspace sampling approach) achieves better running times than SWCC as the number of dimensions increases. However, the values in Table 5.9 suggests that the growth rate of SCBS is faster than that of SWCC. Specifically, with $d = 100$, SCBS is 11.5 times faster than SWCC (2.89 minutes versus 33.44 minutes); however this ratio is reduced to 2.3 as the number of dimensions increases to 3500. This is consistent with the sub-quadratic time complexity (w.r.t the dimensionality) of SCBS compared to the linear time complexity of SWCC. SSC completes the experiment with $d = 100$ in 39.37 minutes and raises a memory error for higher values of d. Table 5.9 also provides some insights on the complexity of the algorithms. SWCC is linear to the number of dimensions and data points [50] and it can be observed that the running times of SWCC on the $10,000 \times 100$ dataset (Table 5.9) is approximately ten times of its running time on the $10,000 \times 10$ dataset in the previous experiment (Table 5.8). The only successful run of SCC also shows the smallest increases of running times w.r.t the dimensions: as the number of dimensions increases tenfold from 10 to 100, the running time increases only by six times (6.9 minutes compared to 39.37 minutes). According to Ehsan et al. [76], the time complexity of SSC grows exponentially to the *dimensionalities of subspaces*, which we keep constant at 20 in this experiment, which partially explains why such exponential growth of running time is not observed.

The running times in Table 5.9 show that by using the subspace sampling approach, the proposed algorithm can cluster a large dataset having 10,000 data points and up to 3,500 dimensions in around 10 hours. Although the long running times can be partially accounted

for by the large volume of the dataset, it is important that the sampling approach allows the algorithm to keep its computational complexity sub-quadratic to the dimensionality of the input, as shown in Figure 5.7. This overcomes the challenge of the quadratic complexity with respect to the number of dimensions and allows the algorithm to scale better to high dimensional datasets. More specifically, the running time has the expected complexity of $O(d\sqrt{d})$ in this experiment. We further analyze the running times by formulating the function of running times with respect to the dimensionality. The fitted function has the form of $time = 0.003 \times d\sqrt{d} - 3.32$ (minutes), and is plotted in Figure 5.7.

Although the times grow sub-quadratically to the number of dimensions, we observe the constant is still quite high. Here, the constant is 0.003 minutes (0.18s), and the function suggests that the algorithm will take more than 17 hours to cluster a dataset of size $10,000 \times 5,000$. The proposed subspace sampling approach indeed pushes the scalability further and allows our algorithm to scale from datasets having hundreds dimensions to datasets that have thousands dimensions. This widens the range of practical applications that our proposed algorithm can be applied to. For example, it can be used to cluster IoT data collected from thousands of sensors, e.g., the pedestrian counting system of the City of Melbourne [1] has 62 sensors (as of December 2019), while the monitored on-street parking network of Melbourne has 4300 in-ground sensors collected at 15 minutes intervals. The algorithm can also be used to cluster gene expressions datasets, e.g., the ten popular datasets [50, 71] have up to 10,000 dimensions but with significantly smaller volumes. Nevertheless, there are many other large datasets with higher or much higher dimensionalities that our proposed algorithm is not capable of clustering. These can include the San Francisco parking data [7] that have 8,200 sensors updated at high frequencies, or textual data that have tens of thousands of dimensions. Other techniques and strategies are needed to push the scalability to dimensionalities even higher, which is discussed in greater detail in the future work discussion in Chapter 7.

5.4.6 Summary of Evaluation

We summarize the experiments and the key findings of our evaluation in this section.

- We generate synthetic datasets that have different properties to evaluate our proposed clustering algorithm on its capability in (1) finding clusters in disjoint and non-disjoint subspaces, (2) clustering data having different levels of outliers, and (3) scaling to datasets that have large volumes and high dimensionalities.

- Our algorithm SCBS produces comparable or better results than SSC and SWCC, and outperforms other baseline algorithms. We also observe that SCBS produces consistent clustering results for a wide range of parameters k.

Fig. 5.7 Running times of the algorithm when clustering datasets that have 10,000 data points with dimensions in range $[100, 3500]$.

- SCBS outperforms other algorithms in clustering data that has outliers. As the levels of outliers increase from 0% to 60%, the metrics NMI, F1 Score and RI decrease at the slowest rate among all other algorithms and are still sufficiently high to indicate good clustering quality.
- The evaluation indicates that SCBS is able to discover clusters that reside in non-disjoint subspaces.
- We demonstrate the SCBS is able to scale to datasets that have large volumes and high dimensionalities. Specifically, it clusters data that has up to one million data points, or data that has up to 3500 dimensions, faster than other algorithms and within reasonable running times.

5.5 Summary

We have presented a subspace clustering algorithm that is able to find clusters in non-disjoint subspaces. The algorithm leverages our proposed technique of constructing the similarity in high dimensional space by aggregating similarities in lower dimensional base subspaces. The main challenge lies in addressing the expensive computation and maintenance of a set of locally relevant subspaces for each pair of points, as well as in aggregating multiple

base similarities *for different groups of points*, instead of just considering point-to-point relationships. To this end, our two phase algorithm *SCBS* searches for low dimensional similarities by clustering data points in low dimensional base subspaces, then subsequently adopts the frequent pattern mining technique to aggregate these base clusters into high dimensional subspace clusters. We then analyze the time complexity of the algorithm and address its limitations in scaling with the high dimensionalities by proposing a derived version of the algorithm that uses a subspace sampling technique. This sampling technique enforces on the minimum frequency that each dimension needs to be assessed. We have also proven in Lemmas 4.3.2 and Lemma 4.3.3 that sampling can still ensure similarities in high dimensional space. We thoroughly evaluate the proposed algorithm on its abilities to cluster data in disjoint and non-disjoint subspaces with data having different volumes, dimensionalities, and levels of outliers. The algorithm using the subspace sampling technique is also evaluated in comparison with the original version and is shown to give comparable results for considerably lower running times. We also evaluate its scalability on the volume as well as the dimensionality of the input. We show that it can cluster datasets having up to one million data points, and large datasets having up to 3,500 dimensions with reasonable running times. This makes the algorithm practical to cluster large datasets generated by various applications in real life. In the next chapter, we investigate the ability of our proposed algorithm to find clusters in three real-life applications.

Chapter 6

Applications

In this chapter, we evaluate the ability of our proposed algorithm to find clusters in datasets produced by different real-life applications. First, we revisit the problem of modeling the pedestrian distribution of the City of Melbourne. We demonstrate that, by addressing the research challenges raised in Section 2.2, we can successfully cluster the pedestrian counting data in higher dimensions, which our previous attempt failed to achieve in Chapter 3. Specifically, we cluster data generated by sensors deployed at 43 locations in the CBD of the City of Melbourne, and find clusters that correspond to pedestrian distributions at different times of the day. We also show how the algorithm can be used to analyze the impacts of a major change in public transport to the activities of pedestrians. Second, we demonstrate that the proposed algorithm is capable of clustering much higher dimensional data. Specifically, we apply our subspace clustering algorithm to ten gene expression datasets, where the number of dimensions grows up to 10,000. Next, we demonstrate how SCBS can be used as an intermediate step to assist more complicated decision making processes. In particular, we use the clustering result to build an ensemble classification model and show that it improves the accuracy in predicting the car parking occupancy of 276 parking bays in the CBD of the City of Melbourne.

The applications discussed in this chapter have been presented in the following papers: M. T. Doan, J. Qi, S. Rajasegarar, and C. Leckie. Scalable Bottom-up Subspace Clustering using FP-Trees for High Dimensional Data. Proceedings of the 6^{th} IEEE International Conference on Big Data (IEEE BigData), 2018

6.1 Analyzing Urban Pedestrian Activity in the City of Melbourne

In this section we explore two applications of SCBS on the data generated by the City of Melbourne Pedestrian Counting System [1]. First, we address the main limitations of our analysis in Chapter 3: we were only able to analyze data of weekdays, having up to 10 dimensions, even with the use of categorized data. To this end, we perform the same task of profiling different types of pedestrian distributions using SCBS. Specifically, we aim to show that our clustering algorithm is now able to cluster data collected at all 43 locations in the CBD and over all days of the week, and produce meaningful insights on pedestrian activities. The clustering algorithm is performed on batches of two month's worth of data, from Feb 2016 to Mar 2017. The clustering result is evaluated both qualitatively and quantitatively to validate the clusters produced by the algorithm.

In the second application, we show that the clustering result of SCBS is able to reveal the contrasting distributions of pedestrians due to the Night Network, which is a major change to the public transport of the City of Melbourne. Specifically, we perform clustering on data collected around midnights for weekends from Jan to Sep of 2015 and 2016, then validate the results not only subjectively using our understanding of the pedestrian activities, but also by using the technique of iVAT and contrast mining [72].

6.1.1 Modeling Pedestrian Distributions

Let X be a collection of N pedestrian counts from time t_1 to t_N: $X = \{x_i : i = 1..N\}$, where x_i corresponds to the pedestrian counting values observed at time t_i. Let $X_j \subset X$ be a subset of X that consists of the pedestrian counts at certain times during the observed period. Let L denote the full dimensional space of the data, $L = \{l\}$ where each dimension l corresponds to a location where the data are collected. Let $L_k \subset L$ be a subset of L that consists of dimensions that correspond to a group of locations.

We aim to find the set of c clusters $\mathscr{C} = \{C_{L_k}^{X_j}\}$, where each cluster $C_{L_k}^{X_j}$ contains observations X_j captured at times $\{t_j\}$ at locations $l \in L_k$ and represents a pattern of pedestrian distribution. We then analyze both the time and location components of each cluster and perform two types of verification on the clustering results:

- First, our qualitative evaluation interprets the times and locations of each cluster to justify the cluster validity based on our understanding of pedestrian activities in the City of Melbourne.

- Second, we perform quantitative evaluation on the grouping of the times of the days in each cluster, and the grouping of locations in each subspace. Specifically, we re-use the ground truth of the types of distributions of pedestrians that we propose in Chapter 3 (with one additional label that was not covered in the experiment in Chapter 3) to quantify the homogeneity of the types of pedestrian distributions being grouped in each cluster. More details of the evaluation method, and the results are presented in the following sections.

We apply our clustering algorithm to analyze the pedestrian count data collected from 31^{st} January 2016 (Sunday) to 1^{st} April 2017 (Saturday) by sensors deployed at 43 locations in the CBD. The pedestrian distribution at these locations, including major train stations, shopping malls, office areas, tourist attractions, and bar and restaurant areas, represent the flows of pedestrians within the CBD and reflect different activities of the city. The start and end date was selected so that the period starts on Sunday and ends on Saturday, and we have equal frequencies of each day of the week. The first month of January is excluded since it contains multiple abnormal events of the year, including low activities in the first two weeks of the new year, and Australia Day. We also show that compared to our approach in Chapter 3, SCBS is able to cluster higher dimensional data but it is also more robust, by applying the algorithm on the data collected on every day of the week (note that we excluded weekends in the experiment in Chapter 3). In summary, the data being analyzed covers 14 months, and consists of 10,224 data points, each of which corresponds to the pedestrian counts collected at 43 locations around the CBD at a specific time of the day.

In this section, each iteration of clustering is performed on two months' worth of data. The reason for using a short period of data in clustering is to address the nonstationarity of the data. In Chapter 3, we observe that pedestrian activities are dynamic and change intermittently. We also demonstrated the benefits of clustering shorter windows of data in Chapter 3 by showing the superior performance of the dynamic Ensemble Switching Model compared to the once-off HyCARCE algorithm. Various factors can affect pedestrian distributions, such as holiday seasons, weather, and special events, which can vary significantly from month to month in Melbourne. For example, the much lower temperature in July together with the earlier sunset time compared to November lead to lower pedestrian counts at the same evening time in July. In fact, the average count of pedestrians at Princes Bridge at 7pm on Thursdays in July is at the same level of the average count of pedestrians at 9pm on the same days in November. For this reason, clustering a long period of data can lead to different hours of the day being grouped to the same clusters due to nonstationarity, which might still be valid but nevertheless makes the evaluation more challenging. Here, by considering a short window of two months, we make the assumption that the system of pedestrian distributions

does not switch between states, and each hour of the day reflects only one type of pedestrian distribution.

We present a qualitative evaluation by interpreting the clustering result on the data collected from 1^{st} June 2016 to 31^{st} July 2016 (a dataset of 1440 data points that have 43 dimensions) in Table 6.1. Similar to the presentation in Chapter 3, in order to keep the table concise and not having multiple small groups, we only show components whose frequency is larger than 30% of the input volume, i.e., we only display components that appear more than 18 times (30% $\times$ 60 days) in a cluster. Note that this is only for presentation, and our quantitative evaluation in the next section uses the entire set of clustering results.

Qualitative Evaluation

Table 6.1 Clustering result of June to July 2016. The instance counts on each day of the week are computed on the cluster components shown in the table only (and not for the low frequency components). Abbreviations: FS - Flinders Street, MC - Melbourne Central, CS - Collins St, TH - Town Hall, BS - Bourke Street, CT - Chinatown, SB - SouthBank, CP - Collins Place, LS - Lygon Street, WF - WaterFront, BM - Birrarung Marr, NQ - New Quay, FStaff - FlagStaff, SC - Southern Cross, LD - Lonsdale Street. Full details of sensor locations are available in Table A.2 in the Appendix.

Clusters	Partitions of points								Subspaces
	(Clusters components, Instance Count)	(Days of observations, Instance Count)							
		Sun	M	T	W	T	F	S	
C1	(7, 25)	0	7	6	2	7	2	1	FS - Swanston Street, MC, TH, CS
C2	(8,31)	0	5	5	8	6	6	1	
C3	(9,33)	0	6	7	6	7	7	0	FS - Swanston Street, MC, TH, CS, CP - North, BS - North/South
C4	(10,21), (11,19), (14,23), (15,27)	2	22	18	19	21	6	3	CT - North/South, MC, SB CL - North/South
C5	(12,28), (13,23)	1	8	9	13	11	8	1	CT - North/South, MC, SB CL - North/South LS - East/West
C6	(16,32)	0	6	5	6	7	8	0	WF, BM, BS - North/South, LS - East/West, NQ, CT - North/South
C7	(17,22)	0	4	5	4	8	0	1	FS - Swanston Street, MC, FStaff
C8	(18,27)	0	6	7	6	8	0	0	LS - East/West, NQ, CT - North/South
C9	(19,32), (20,28), (21,34), (22,31), (23,36), (midnight,33)	33	17	40	38	23	0	38	All locations in the CBD except SB, FS - Swanston Street, CT - North
	(1,43), (2,41), (3,35), (4,36), (5,44), 6,39)	34	35	36	36	38	30	27	
	(7am-7pm, 75)	24	0	0	0	0	0	37	

SCBS finds 9 clusters from the data. The details of each cluster are described as follows:

- Clusters C1 and C2 each corresponds to an hour during the morning peak from 7am to 9am, mostly on weekdays (there is only one instance that belongs to Sat). These clusters express higher counts of pedestrians than the two-month average counts across

all locations represented by the dimensions of the subspaces. The locations that are grouped in the subspaces of these clusters include the main entrance of two major train stations (Flinders Street station and Melbourne Central station), and two other locations on Swanston Street (Town Hall and City Square). These locations are usually crowded with pedestrians on their way to offices, university, restaurants and cafeterias in the morning.

- Cluster C3 corresponds to 9am of the morning peak. While the subspace of this cluster also has the same 4 locations as C1 and C2, the correlations between these locations are different. In each of the 2*D* base subspace that is constituted by locations {FS - Swanston Street, MC, TH, CS}, the base clusters of cluster C3 are different from the base clusters of C1 and C2. Further inspection shows that the average values of pedestrian counts at 9am increase at Town Hall, and decrease at the other three locations. Cluster C3 also groups three additional locations, which are CP - North (office area), and BS - North and BS - South (shopping mall area). The inclusion of these three locations suggests a pattern of pedestrian activities around office and shopping mall areas that are not observed at 7am and 8am. While clusters C1, C2 and C3 agree with our understanding of pedestrian activities in Melbourne during the morning peak, we also note the lack of inclusion of other areas of the CBD, especially the other two major train stations in the CBD that are also monitored.

- Cluster C6 describes the pedestrian distributions at 4pm, before the afternoon peak. The subspace of cluster C6 corresponds to tourist attractions (Waterfront City, Birrarung Marr), shopping areas (Bourke Street mall North/South, Chinatown), and restaurant areas (Lygon Street, New Quay, Chinatown). Here, the average values of the observations in the cluster indicates that pedestrian counts are low at tourist attractions and restaurant areas (except for Chinatown), and high at Bourke Street shopping mall areas.

- Towards 5pm and 6pm, the correlations are observed more at the major train stations, and nearby restaurants around the centers of the CBD, as expressed by clusters C7 and C8. More specifically, the observations of these clusters show increasing pedestrian activities at these areas at 5pm and 6pm from Mondays to Thursdays. Note that although C7 and C8 share the same subspace, their base clusters in the 2*D* base subspaces are not all similar (hence, the final output is two separate frequent patterns). This suggests that the activities at 5pm and 6pm are still distinctive at these locations. By analyzing the raw values of these instances, we observe that, from 5pm to 6pm, the pedestrian counts increase at Lygon Street (North and South), Chinatown, and decrease at the 3 major train stations. Clusters C7 and C8 also correctly identify the difference

in the pedestrian distribution at 5pm-6pm on Fridays compared to other days of the week, and exclude the observations of Fridays from the clusters. This is consistent with our understanding of the pedestrian behavior on Friday afternoons, where more people usually spend a longer time in the CBD before the weekend.

- C9 describes a low level profile of pedestrian distributions that are observed during the evening after 7pm, and early mornings (1am-6am) every day of the week. It also identifies that the levels of the pedestrian activities in the CBD during weekend mornings and afternoons are low, and groups them into this cluster. It should also be noted that the observations on Friday nights are selectively excluded from the cluster, for the same reason provided above. Analyzing the subspace of this cluster reveals that the patterns are observed across almost all the locations in the CBD during these odd hours, except for Southbank, Flinders Street - Swanston Street, and Chinatown North. The common attribute that these locations share is that they are surrounded with various bars and restaurants that are open till late in the evening, which can result in higher pedestrian activities during these hours, compared to the rest of the CBD. This explains the absence of these locations in the cluster subspace.

In summary, the clustering result is valid and consists of clusters that correspond to different types of pedestrians at different times of the day. This example also demonstrates the algorithm's ability to discover clusters in non-disjoint subspaces. For example, it discovers that there are correlations between {Lygon Street East, Lygon Street West, New Quay, Chinatown} at 5pm and 6pm, and that the correlations are different and belong to different clusters (C6, C7, C8). Another example of the non-disjoint subspace is the overlapping subspace between C1, C2 and C3.

Nevertheless, we also identify the limitations in the cluster coverage in this experiment as follows. First, all clusters cover only 68% of all the data points (455 out of 1440 data points do not belong to any clusters). The days whose observations are least covered by any cluster are Fridays, Sundays, and Monday. The hours whose observations are least covered are 7am, 11am, 1pm-2pm, 7pm-10pm. Second, except for the large cluster C9 that describes the very low distributions of pedestrians during midnight to early morning, and during weekends, the subspaces of other clusters are constituted from only 18 out of 43 locations. That is, *no other patterns are discovered for the other 25 locations during other times of the day*. These results raise the question of whether all relevant patterns and subspaces have been identified. In the next section, we aim to further quantify this, along with the algorithm's overall performance, by evaluating the results with our proposed ground truth.

Table 6.2 Labels of hours of the day.

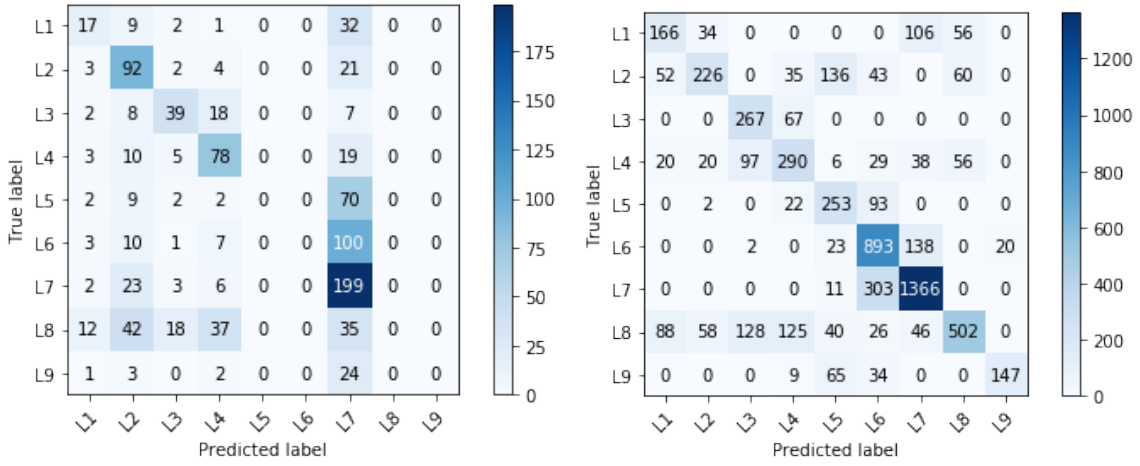
Description	Hours	Day of Week	Label
Early morning	6am, 7am	Mon-Fri	L1
Morning peak	8am, 9am	Mon-Fri	L2
Afternoon peak	5pm, 6pm	Mon-Fri	
Lunch time	11am-1pm	Mon-Fri	L3
Quieter hours between peaks	10am, 2pm-4pm	Mon-Fri	L4
Evening time	7pm-9pm	Mon-Fri	L5
Late evening time	10pm-midnight	Every day	L6
Midnight to early morning	1am-5am	Every day	L7
Weekend - day	6am-6pm	Sat-Sun	L8
Weekend - night	7pm-9pm	Sat-Sun	L9

Quantitative Evaluation on the Clustering of Points

We first quantitatively evaluate the clustering of points (rows) by measuring *to what extent each cluster can express distinguished characteristics of pedestrian distributions at a certain time of the day*. To this end, we re-use the ground truth we first proposed in Chapter 3, which labels each hour of the day to a specific type of pedestrian distribution. The proposal of the ground truth was first presented in Section 3.3.4 and Table 3.5. Note that we add additional labels L8 and L9 and adjust labels L6 and L7 to accommodate for the weekends that were not previously considered in Chapter 3. The ground truth is presented again in Table 6.2 for easy reference.

The ultimate label of a cluster is derived by a popularity vote of the labels of its data points (each data point corresponds to an hour when the pedestrian count is captured). For example, the majority of cluster C1's data points are records collected at 7am, hence its predicted label is L1 (a more detailed working example is presented in Table 3.6). Subsequently, all observations of clusters C1 are assigned to label L1, while their true labels are decided by the actual hour of each observation based on Table 6.2. We present the confusion matrix for the period of Jun-Jul in Figure 6.1a, and the confusion matrix for the entire 14 months in Figure 6.1b. Note that the confusion matrices are computed only on the data points that are clusters, e.g., 455 data points do not belong to any cluster for the period Jun-Jul 2016, so they cannot be assigned to any label. Therefore, the confusion matrix contains only $1440 - 455 = 985$ instances.

Figure 6.1a shows that the most frequently predicted label is L7 (507), which corresponds to the period from after midnight to 5am of the next morning. While the labels of 199 instances are correctly predicted, 100 instances with label L6 and 70 instances with label L5 are misclassified as L7. This can be explained by the very big cluster C9 that groups four



(a) (b)

Fig. 6.1 Confusion matrices of predicting characteristics of pedestrian activities for (a) Jun 2016 - Jul 2016, (b) Feb 2016 - Mar 2017

large sets of hours: 7pm to 9pm (L5), 10pm to midnight (L6), after midnight to 5am (L7) and weekends daytime hours (L8). The popularity vote is dominated by the label L7, therefore the presence of the other three groups are obfuscated. This explains the absence of labels L5, L6, L8 and L9 in the predicted values. The darker diagonal blocks for L1, L2, L3 and L4 suggest that the clustering results have clusters that homogeneously contain the hours that correspond to these labels.

Figure 6.1b summarizes the evaluation result for all 14 months of data. The number of data points is 10,080, among which 6,198 points are grouped to a cluster. Note that the clustering algorithm is applied on each subset of two months' worth of data, due to the non-stationary property of the data as discussed in the previous section. The confusion matrices for each subset are presented in Figure A.3 in the Appendix. Figure 6.1b still shows the darker diagonal blocks. This is a good indication of the homogeneity of the clusters, i.e., each cluster contains observations that agree and distinctively correspond to a single type of pedestrian distribution. The results show that the clustering algorithm performs well in finding clusters corresponding to labels L6, L7, L8 and L9. The precision, recall and F1 scores in predicting these labels are higher than those of other labels. The better performance in predicting these labels can be explained by the relatively more stable counts of pedestrians during these hours of the day, especially during the late evening time (L6) and after midnight to early morning (L7).

While the ability to cluster data of all days of the week can be regarded as an improvement compared to the experiment in Chapter 3 (in which only weekdays are considered), the current

Table 6.3 Metrics of predicting characteristics of pedestrian activities.

Labels	Precision	Recall	F1 Score
L1	0.51	0.46	0.48
L2	0.66	0.41	0.51
L3	0.54	0.80	0.64
L4	0.53	0.52	0.53
L5	0.47	0.68	0.56
L6	0.63	0.83	0.72
L7	0.80	0.81	0.81
L8	0.74	0.50	0.60
L9	0.88	0.57	0.70
Overall (weighted)	0.68	0.66	0.66

ground truth shows that the performance of the algorithm in finding clusters corresponding to weekend activities can still be improved, especially for the low recall in predicting label L8 (0.50). It can also be observed from Figure 6.1b that many instances are predicted to labels L3, L4 and L1 (in the order of the instance counts) while their true label is L8. This explains both the low recall in predicting L8, and low precision in predicting L3 and L4. Based on our understanding of pedestrians in Melbourne, the pedestrian activities are considerably different in most of the locations in the CBD from 6am to 6pm during weekends, especially at the train stations (lower distributions), office areas (lower distributions), and shopping mall areas (higher distributions). By analyzing the clusters that are predicted to be L3 (11am-1pm during weekdays) and L4 (10am, 2pm-4pm during weekdays) while their true label is L8, we discover that the subspaces of the clusters mainly contain dimensions that correspond to locations where the pedestrian counts are relatively stable and are less susceptible to a significant change of pedestrian count between weekdays and weekends, and throughout different times during weekends. For these reasons, these clusters contain some weekday events, and *a higher number of weekend events*, which eventually leads to the cluster being assigned to label L8 by the popularity vote. These locations include the outskirts of the CBD such as Waterfront city, Webb bridge, and Tin Alley-Swanston St, or other places that are normally crowded such as Chinatown - North, State Library, and Bourke Street Mall (North/South). While a better ground truth can help better evaluate this case, it is not a trivial task. Such a ground truth needs to record the fact that some hours during weekdays and weekends can share the same label at some locations (as in the example described above), while they need to have separate labels at some other locations. That leads to a segmentation of the ground truth not only on the times of the day, but also on the locations, which further

increases the complexity of the ground truth. In the scope of this analysis, we consider the reported results as the lower bound of the algorithm's capability in discovering different types of pedestrian distributions.

Quantitative Evaluation on the Grouping of Dimensions

The evaluation of the dimensions of a subspace is not as straightforward. The fundamental differences between this evaluation and the evaluation of the clustering of observations (points) is that while it is expected to have observations of the same cluster to correspond to a type of pedestrian distribution (e.g., morning peak, lunch time, etc.), *there is no guarantee that the types of locations represented by dimensions of a cluster subspace will be homogeneous*. A cluster can correspond to a correlation of pedestrian counts at locations having different natures. For example, cluster C6 in Table 6.1 contains observations where pedestrian counts are high at *shopping areas* (Bourke Street mall), and low at *tourist attractions* and *restaurant/cafe areas*. In this cluster, the data points exhibit the same pattern described above, but the locations represented by the dimensions are different in their nature. Moreover, the non-homogeneity of location types can be observed in all clusters in Table 6.1. For this reason, we cannot use the same approach of taking the popular vote to assign a cluster's dimensions to a single category.

In fact, the non-homogeneity of types of locations in a cluster subspace raises significant challenges in proposing a ground truth for evaluation. According to our understanding of pedestrian activities, there is no constraint on what types of locations should or should not be grouped together in the same cluster. We analyze the dimensions of subspaces of the clusters discovered, and list some examples below:

- One cluster shows pedestrian counts are high across all four train stations being monitored in the CBD.
- One cluster shows pedestrian counts are high at some train stations (Flinders Street and Melbourne Central stations) and low at other train stations (Flagstaff and Southern Cross station). This cluster corresponds to observations around lunch time, and Flinders Street and Melbourne Central stations are surrounded with more restaurants and food stores than the other stations.
- One cluster shows pedestrian counts are high only at tourist attractions (example: at 3-4pm).
- One cluster shows pedestrian counts are high at train stations, offices and restaurant areas, and low at tourist attractions and shopping mall areas (example: morning peaks).

- One cluster shows pedestrian counts are high at office, restaurant and shopping mall areas (example: afternoon peaks).

These examples (there are many more that we do not include) illustrate that there can be multiple valid combinations of location types in a cluster subspace, ranging from a single type, two types, or multiple types of locations. For this reason, proposing a realistic ground truth would require further investigation to understand which locations are correlated at what time of the day, and subsequently should be grouped together in the same subspace. Such analysis should not rely solely on the pedestrian counts but also needs input from urban studies and city planners. For this reason, due to the lack of ground truth, we conduct an internal evaluation on the location components of the cluster subspaces.

Specifically, we quantify the types of locations in each cluster that is discovered. To this end, we first categorize each location of the deployment map into five types: train station, shopping center, restaurant/cafe, tourist attractions and office areas. The categorization is described in Table A.2 in the Appendix and follows this guideline:

- A location is categorized as a train station, shopping center, tourist attraction or office area if any of those points of interest are within 100 meters radius of the location at which the sensor is deployed. For example, St Kilda Rd-Alexandra Gardens is a tourist attraction and is also classified as a train station type of location due to its proximity to Flinder Street station.

- A location is categorized as a restaurant/cafe type only if there are more than 10 restaurants and/or cafes within 100 meters radius of the deployment location. This is to take into account the high distribution of restaurants and cafes around the CBD.

The result of performing clustering on seven two-month periods from Feb 2016 to Mar 2017 contains 95 clusters, corresponding to 95 subspaces (each subspace corresponds to a combination of locations). We analyze the types of locations and summarize the result in Table 6.4.

There are only 10 subspaces that contain all 5 types of locations, 7 of which are subspaces of clusters that correspond to the period of midnight to 5am of the next morning. As described in the previous section with cluster C9 in Table 6.1 as an example, this period usually observes very low and stable counts of pedestrians across the entire CBD. There are only three subspaces that contain a single type of location, all of which belong to train station type. The most common type of cluster subspaces contains three and four types of locations. This observation demonstrates the complexity of the patterns of pedestrian distributions, and shows that pedestrian activities in different types of locations tend to be correlated.

Table 6.4 Summary of types of locations of cluster subspaces.

Number of distinct types of locations	Number of subspaces	Percentage
5	10	10.5%
4	25	26.3%
3	46	48.4%
2	11	11.6%
1	3	3.2%
Total	95	100%

We identify the coverage of the clustering result as a limitation of this experiment. Specifically, all the clusters discovered by the algorithm cover only 61.5% of all the data points. The top five hours that are least covered by a cluster are 5pm, 4pm, 11am-12pm, and 8pm. Moreover, the subspaces of clusters do not have high coverage of the dimensions either as described earlier. Specifically, while 95 cluster subspaces are constituted of all 43 dimensions, if we exclude the big clusters that "blanket" nearly all the locations of the CBD during the midnight-early morning hours (e.g., cluster C9 in Table 6.1), the remaining clusters *cover only 35 out of 43 locations*. For this reason, while we believe the clusters discovered by SCBS are valid and describe distributions of pedestrians that agree with our understandings, the algorithm might have only discovered distinctive and emerging patterns, and there could be other patterns or distribution types not yet discovered by the algorithm. Nevertheless, the clustering result can still offer these insights:

- The clustering model creates load profiles of pedestrians at different times of the day and locations in the city. This knowledge is valuable to various audiences. First, it can assist in optimizing the public transport schedule by rebalancing the resources according to pedestrian load profiles, e.g., the model can suggest whether the allocation of trams and buses on a specific route is in accordance with the distribution of pedestrians during a particular time of the day. Second, business managers, knowing the spatio-temporal characteristics of the pedestrian distribution, can make informed decisions to better target customers, and forecast the required resources for their business. Moreover, the model presents itself as a reliable reference for city planners to evaluate and assess the impacts of any changes introduced to the city. The changes might include public events, construction, or changes in policies or the public transport system. We present a use case of our clustering model to evaluate the impact of a major change in public transport introduced to the City of Melbourne in the next section.

- Analyzing the clusters that correspond to continuous hours can offer insights into the flow and movement of pedestrians (an example is provided in the qualitative evaluation on the clustering result in Table 6.1).

Summary of Modeling Pedestrian Distributions

In this experiment, we apply SCBS on the pedestrian count data to profile different types of pedestrian distributions. This experiment addresses the limitations of our work in Chapter 3 by expanding the analysis to include data captured during weekends and at more locations.

We first perform qualitative evaluations by investigating each cluster that SCBS discovers, both in terms of the times (data points) and locations (dimensions). We conclude that the clustering result is valid and consistent with our understanding on pedestrian activities in the CBD of Melbourne. We also observe the problem of limited coverage of the subspace clusters on the data points as well as locations, which is subsequently discussed in our quantitative evaluation.

We quantify the performance of our result on the clustering of data points and the grouping of dimensions. First, we use the same ground truth that was proposed in Chapter 3 and observe an improvement in the results, as reflected in the recall, precision, and F1 score. We also discuss in detail the performance of SCBS in finding clusters that correspond to types of pedestrian distributions that correspond to different times of the day. Next, we classify the locations into five location categories in order to quantitatively evaluate the grouping of locations.

6.1.2 Analyzing Impacts of the Night Network

In this section we discuss a practical application of our clustering model to evaluate the impact of a major change in the public transport system on the activities of pedestrians. Since January 2016, Public Transport of Victoria (PTV) has trialed a new program that provides all-night public transport options for over 70% of Melbournians during weekends [4]. The options include metropolitan and regional trains, late night buses and trams to and from the CBD. The trial has been closely monitored and assessed on multiple aspects. Our aim is to provide an analysis on how the new transport program affects pedestrian activities in the CBD. Note that the analysis in this section and its results were used by PTV in their analysis of the impact of this 24-hour transport trial.

Prior to the trial, all public transport stopped at midnight on Fridays and Saturdays. For this reason, we focus our analysis on the patterns of pedestrians at midnight (0am), 1am and 2am of Saturdays and Sundays in 2015 and 2016. Since the roll out of the trial, we had 9

Table 6.5 Clustering results of 468 observations during the midnight period of 2015 and 2016 from January to September.

Clusters	Breakdown of data points of cluster (year) midnight	1am	2am	Dimensions (locations) of subspaces
C1	55 (2015) 61 (2016)	5 (2015) 3 (2016)	3 (2015) 6 (2016)	
C2	13 (2015) 2 (2016)	53 (2015) 8 (2016)	52 (2015) 12 (2016)	Flinders Street-Swanston Street, Princes Bridge, the Arts Centre Melbourne Central, Flagstaff station, Southern Cross station Chinatown (South/North), Town Hall
C3	2 (2015) 7 (2016)	12 (2015) 43 (2016)	9 (2015) 44 (2016)	Flinders Street-Swanston Street, Princes Bridge, the Arts Centre Melbourne Central, Flagstaff station, Southern Cross station Bourke Street mall (South/North)
Outliers	30 (2015) 48 (2016)			

months' worth of data for analysis. To this end, we extracted the pedestrian numbers observed at those hours (midnight to 2am) on Saturdays and Sundays from January to September in both years, resulting in 39 weekends being analyzed. The final dataset consists of 468 observations (3 hours $\times$ 2 days $\times$ 39 weekends $\times$ 2 years). We first apply SCBS on the dataset and find 3 clusters as described in Table 6.5.

SCBS finds three clusters. As shown in Table 6.5, the first cluster C1 contains 74% of observations collected at midnight in both years. The cluster C2 contains 67% of observations collected at 1am and 2am of 2015, and cluster C3 contains 56% of observations collected at 1am and 2am of 2016. This clustering result, in addition to suggesting different pedestrian distributions from the midnight to 1-2am periods, also shows that the pedestrian distributions during the period of 1am-2am are sufficiently different from the year 2015 to 2015 to form two separate clusters.

The two clusters C2 and C3 show substantial overlap in the dimensions that correspond to locations at and around all four major train stations. The difference between the two dimension sets include Town Hall, Chinatown, Bourke Street Mall. In relation to late night activities around these areas, we provide the following possible explanations:

- Chinatown has multiple bars and restaurants that are open past midnight.
- Town Hall on Swanston Street is surrounded with multiple bars that are open late. In addition, Town Hall is adjacent to the intersection of Swanston Street and Flinders Street, which has a large taxi hub, and is the nearest option for taxis (normal vehicles are prohibited along Swanston Street).
- Although the Bourke Street Mall is a busy area during the day, almost all offices, shops, bars, and restaurants close well before midnight. Bourke Street Mall is co-located with

one of the tram stops of tram routes 86 and 96, whose services are included in the Night Network program. These tram routes provide new after-hour transport options to commuters from the CBD to the South and East of Melbourne. We believe that the late hour tram commuters at this stop provide a plausible explanation for the inclusion of the location at Bourke Street Mall in the cluster subspace for only the cluster of year 2016.

Next, *guided by the clustering result*, we attempt to visualize the difference in pedestrian distributions before and after the changes. To this end, we keep only the locations identified by the cluster subspace and use iVAT to visualize the cluster structure. iVAT is a popular visual method for determining the number of clusters, or the cluster tendency, of a set of data objects. The method works on a pairwise distance matrix of a set of data objects. It reorders the objects such that similar ones (small distance) are near each other in the dataset. Subsequently, the pairwise distance matrix of the reordered data is displayed as an intensity image, in which the hidden cluster structure is revealed as dark blocks along the diagonal. The algorithm of iVAT (and its variations), as well as their applications, are discussed in multiple studies [99, 111, 166].

Naturally iVAT relies on the similarity measures between data points and by default does not work effectively with high dimensional data for the same reason of local feature relevance. Here, we make use of the clustering result of SCBS to learn the information of relevant dimensions and project the dimensional space of the data to only those dimensions. Subsequently, the distances between data points computed in the projected subspaces are guaranteed to reflect the similarities when they are passed to iVAT. We also highlight that without the information on the relevant subspace, this task of visualisation is non-trivial, as we elaborate later in Figure 6.2f.

In order to show the contrast hour to hour between 2015 and 2016, we now perform the visualization for each hour (midnight, 1am, 2am) separately. We first present and discuss the visualization of the difference in pedestrian distributions at 1am.

Figure 6.2a represents the dissimilarity matrix of all observations at 1am, where similar intensities of gray express comparable similarities between observations (darker shades correspond to higher similarities). By having the similarities computed in only the subspace that SBCS deems relevant, Figure 6.2a already reveals two blocks corresponding to two distinct pedestrian distributions in 2015 and 2016 at 1am. Specifically, the first diagonal block for points $\{x_i\}_{i=1}^{3}9$ indicate that these points of year 2015 are more similar to themselves than to those of year 2016. Similarly, in the relevant subspaces identified by SCBS, the second darker diagonal block indicates that the observations of year 2016 are more similar to each other.

Subsequently, we use iVAT to enhance the visualization. iVAT can serve two purposes in this case:

- First, it further amplifies the difference between two years and clarifies the contrast in the visualization if two distinct types of pedestrian distributions indeed exist in the data.
- Second, by having iVAT rearrange the observations, the end result can be used to partially verify the grouping of data points of the clustering result. Specifically, it is expected that the first dark block of iVAT contains mainly the first 39 observations (the observations of 2015), while the second dark block mostly contains the observations of 2016.

The iVAT results in Figure 6.2d shows a much clearer two-diagonal-block structure in the data in the reduced dimensional space. After being rearranged by iVAT, the first block contains 35 data points, 29 of which are observations of 2015 (83%). The second block contains 43 data points overall, 37 of which are observations of 2016 (86%). In addition to the darker diagonal blocks, two off diagonal blocks also indicate that the first 35 data points are significantly more similar to the remaining 43 points. These observations show that there are indeed two different types of pedestrian distributions that exist in the data, and each type of distribution is mostly observed in one year. This result is consistent with clusters C2 and C3 in Table 6.5. This result indicates that there is a change in pedestrian activities from 2015 to 2016 at 1am in the CBD. The visualization of the dissimilarity matrix and the iVAT of pedestrian counts collected at 2am in Figure 6.2b and 6.2e also suggest a change in pedestrian activities between 2015 and 2016 at this hour.

On the other hand, the iVAT image of data collected at midnight in Figure 6.2c does not show any clear distinction of blocks in the data. It might also be arguable that there can be two very light large blocks, besides a slightly darker but much smaller block, that exist along the diagonal. We follow the standard interpretation of the authors of iVAT [99, 179] and conclude that this image does not show any clear cluster structure that exists in the data. We also provide some examples of iVAT, together with the interpretation of the authors, in Figure A.2 in the Appendix for comparison. The iVAT image shown in Figure 6.2c is consistent with cluster C1 in Table 6.5 that contains mainly observations at midnight.

Note that this result of visualization is guided by the information of relevant subspaces discovered by the SCBS clustering result. We next demonstrate that it remains a non-trivial task without this information. Figure 6.2f shows the iVAT image of data collected at 2am, but this time the dissimilarity matrix is computed on the full dimensional data space. Although some darker blocks are still observed along the diagonal, multiple blocks are incrementally

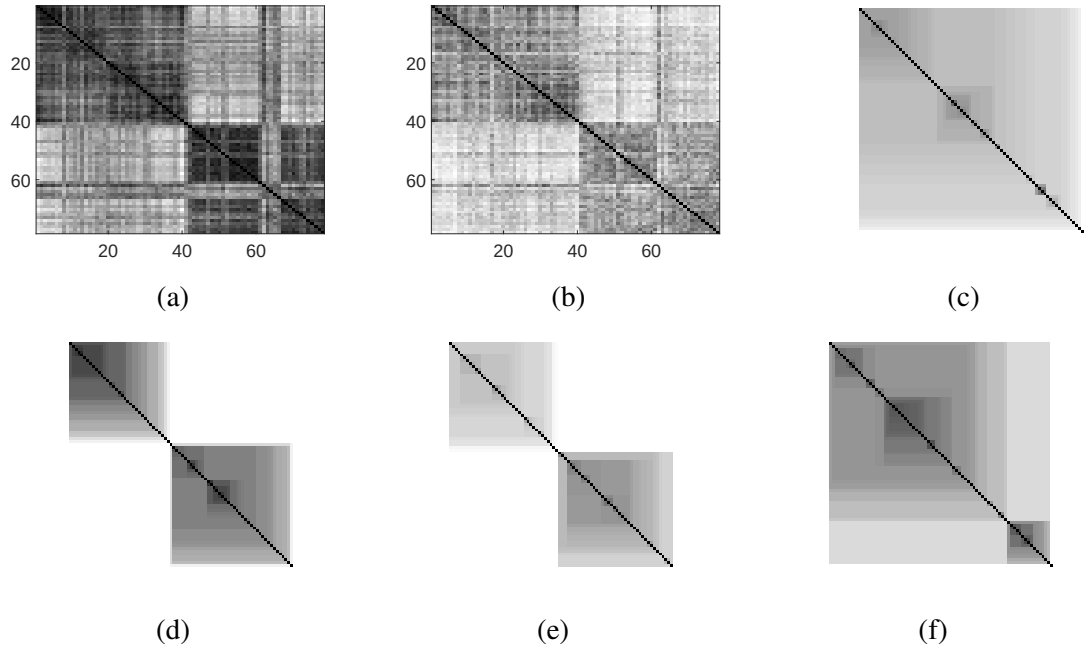


Fig. 6.2 Clustering results of pedestrian counts at different hours: (a) Similarity matrix of observations at 1am showing two distinct blocks, (b) Similarity matrix of observations at 2am showing two distinct blocks, (c) iVAT image at midnight with a relatively uniform intensity, (d) iVAT image at 1am showing two distinct blocks, (e) iVAT image at 2am showing two distinct blocks, (f) iVAT image when applied to the full dimensional dataset captured at 2am showing no clear cluster structure.

overlapping and there are also no white off-diagonal blocks. In addition, further inspection of the blocks reveal that each block does not homogeneously contain the observation of a single year. These indicate that the observations are not very distinctive and the structure of two separate iVAT blocks that represent two types of pedestrian distributions is not detected when all dimensions are considered.

In summary, our clustering result finds three separate clusters. The result indicates that the pedestrian activities at midnight remain the same while the pedestrian distributions at 1am and 2am changed in 2016 after the introduction of the Night Network trial. The cluster subspaces provide information on the locations at which these changes are observed. We subsequently use iVAT to leverage this information to visualize the changes in pedestrian distributions and further confirm the clustering result.

However, the analysis above does not provide insights on the actual changes (e.g., whether the number of pedestrians increases or decreases). We conduct our next experiment using a simplified technique of contrast mining [72] that not only serves as an additional verification of the result of this section but also provides more insights on the actual changes of pedestrian activities after the Night Network trial was introduced.

Verification of Clustering Result on Pedestrian Distribution Using Contrast mining

In this application, we build a contrast mining model to represent the impacts of the Night Network trial on the numbers of pedestrians at each location. First, we aim to use the model to verify whether there are indeed two contrasting patterns of pedestrians before and after the change in the transport system as our clustering result and iVAT suggest. Second, we extract the locations where the changes occur to cross validate with the subspaces of clusters found by our algorithm. To this end, the data used for this analysis contains pedestrian counts collected at 1am and 2am on Saturdays and Sundays, from January to September of 2015 and 2016. In addition to the above, by studying the actual changes of pedestrian counts and mapping them to the geographical map of the City of Melbourne, we explain the shift in pedestrian activities that are introduced by the Night Network.

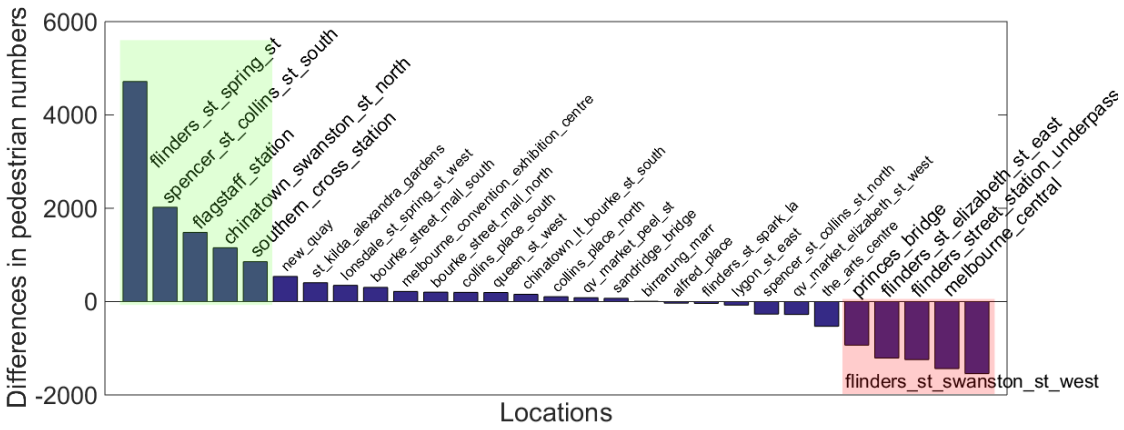
Our method can be summarized in three steps as follows:

- Step 1: Obtain the hourly average of pedestrian counts at each location. We compute a set of $S_i^{(2016)}(t)$ values, which are the values captured at location i at time t of the day in 2016. Therefore, for each location we have one corresponding vector containing the average pedestrian numbers at the hours of interest. We also compute all the values $S_i^{(2015)}(t)$ for the year 2015. Guided by the result of our clustering model, we focus on the time t of 1am and 2am.

- Step 2: Quantify the difference δ_i between two vectors $S_i^{(2016)}$ and $S_i^{(2015)}$ to a scalar value to be used in the next step. Depending on the requirements, different techniques can be applied to calculate δ_i, such as correlation or Euclidean distance. In this study, since the vectors have only two components, and we focus on the changes in pedestrian distribution, we define δ_i as the difference between the total numbers of pedestrians at location i before and after the introduction of the Night Network.

- Step 3: Sort the locations according to δ_i. Identifying the two tails of the distribution reveals the locations at which the pedestrian numbers significantly increase or decrease, respectively. In general, this step can be done in a systematic fashion by identifying the two-tailed Pareto distribution [20]. However, we perform this task by visually inspecting the results in this experiment since the number of locations is not sufficiently high to be statistically significant

Figure 6.3a presents the final result. We also identify the locations where notable changes in pedestrian numbers are observed. The green box highlights the locations that see a significant increase in pedestrian numbers after the Night Network is introduced, while the red box highlights the locations with a decrease in pedestrian numbers. Note that for brevity,



(a)

(b)

Fig. 6.3 (a) Locations sorted by the changes in pedestrian numbers at 1am and 2am. Green box indicates locations that observe significant increases in pedestrian counts, red box indicates locations with significant decrease. (b) Locations with significant changes in pedestrian counts on the map of the Melbourne CBD.

we do not show the locations where the absolute change is negligible. The actual changes in pedestrian counts before and after the change are plotted in Figure A.1 in the Appendix. We pick the top five locations at which the numbers of pedestrians significantly change and offer our hypothesis and explanation. The locations that observe an increase in the number of pedestrians are:

- Southern Cross Station and Spencer Street-Collins Street intersection: Although this major train station is not included in the trial of the Night Network trains (during the trial, Flinders Street station is the only train station in the CBD that is included in the program), these locations are co-located with the hub that services late night *regional train and region coach* (as part of the trial). They are the only available options for commuters to travel to regional areas of Victoria at these hours and appear to result in increased pedestrian activities at these locations.

- Flagstaff station: The location where the sensor is deployed is located near a bus stop that is covered by the Night Network trial. In addition, the location is surrounded with restaurants and bars (we can find at least 8 restaurants and bars that are open till 2am during weekends within the radius of 200 meters of Flagstaff station), and multiple hostels and hotels, which can suggest increased activities of pedestrians around this area.

- Chinatown - Swanston Street (North): This is one of the locations that is most highly populated with restaurants and bars that are open late during weekends (many restaurants are open 24 hours). Similar to the case of Flagstaff station, our hypothesis is that the new transport option of Night Network leads to prolonged pedestrian activities around entertainment areas that were usually terminated earlier otherwise.

- Flinders Street - Spring Street: This location is only moderately populated with bars that are open after midnight (we can only find 3 bars and hotels that are open past 1am during weekends within a radius of 200 meters). It is located next to tram stop 75, and is within 300 meters of the Jolimont train station (not monitored by the pedestrian counting systems). Both are part of the Night Network trial and can partially explain the increase in pedestrian activities. Nevertheless, without further data, we cannot provide more insights into why this location observes the highest increase in pedestrian counts.

The locations that observe a decrease in the number of pedestrians are:

- Princes Bridge, Flinders Street - Elizabeth Street, Flinders Street - Swanston Street, Flinders Street - Station underpass: These locations mark all entrances to the major train station Flinders Street station, that is the only train station to board Night Network trains in the CBD. To facilitate the discussion, we provide an outline of the station in Figure 6.4. The consistent drop of pedestrian counts at all four entrances indicate a decrease in both the inbound and outbound movement of pedestrians to and from

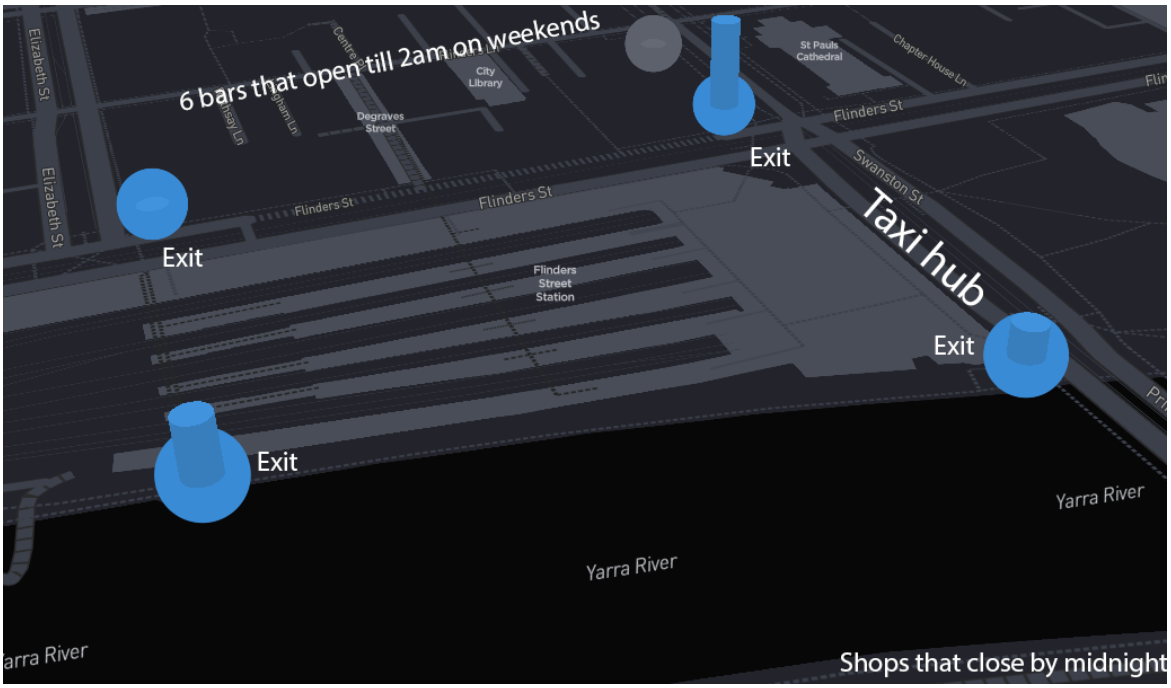


Fig. 6.4 Outline of Flinders Street train station. Unless specified by the notes, all shops that are not mentioned are closed by midnight.

the train station. We offer three explanations for this decrease, despite Flinders Street station being the only departure station for Night Network trains.

- The first factor that contributes to the decrease in the number of pedestrians is the low availability of restaurants and bars that open late. In fact, we can only find, within a 200 meter radius, six bars that open past 2am; most of the shops around Flinders Street station are closed before or at midnight. We believe that restaurants and bars at other parts of the CBD, together with public transport being available at all hours, could have attracted people away from Flinders Street station to better options for late night hangouts.
- The section of Flinders Street station between Princes Bridge and Flinders Street is a taxi hub as marked in Figure 6.4. Our hypothesis is that the new transport option encourages commuters to take public transport, which is a much more economical option. Although the decrease in taxi usage has been verified anecdotally by PTV, we did not manage to acquire any data or statistics on taxi usage to support this hypothesis.

The hypothesis on the decrease in the commuters at Flinders Street station is further supported by the decrease in pedestrian counts in the nearby areas. Out of 11 locations

that observe a drop in pedestrian counts, 7 locations are within 300 meters from the station.

- Melbourne Central station observes a decrease in pedestrian counts although it is a busy hub of late night restaurants and bars (we can identify more than 15 restaurants and bars that open past 2am during weekends). We believe the exact deployment location of the sensor is an important factor that can lead to this phenomenon. Melbourne Central is incorporated into a shopping mall with all the entrances being underground. Therefore, if the sensor is deployed near the underground entrance of the train station, it will not be able to capture all the activities of pedestrians outside the building (the train no longer operates after midnight). Despite our efforts to contact the city council, we were not able to acquire the exact deployment location of the sensor to offer a more reliable explanation. Nevertheless, this is still consistent with the changes observed at Flinders Street station, where a flow of pedestrians moving from two major train stations along Swanston Street to the outskirts of the CBD is observed.

Comparison of locations discovered by SCBS and Contrast Mining

By comparing the two sets of locations where significant changes occur that are discovered by SCBS and our experiment with contrast mining, we observe that both algorithms detect the changes at all four major train stations. While SCBS only reveals the location, the experiment with contrast mining further provides the nature and direction of changes at each of the train stations. The results of the two experiments are also consistent in terms of finding the changes at Chinatown and Princes Bridge.

However, we also observe a higher level of granularity in the reported results of the latter experiment. Our experiment using contrast mining can detect significant changes at locations around the train stations that SCBS missed. Specifically, the changes at Spencer Street - Collins Street and at three exits of Flinders Street station are not detected by SCBS. Note that SCBS is still able to detect the changes at Princes Bridge and the Victoria Arts Center (although this location is not in the top 5).

In addition, the cluster subspaces discovered by SCBS contain some locations that can be considered as not being optimal after inspecting the raw data, guided by Figure 6.3a. Specifically, SCBS detects major changes at Bourke Street Mall (South/North). Although Figure 6.3a and Figure A.1 in the Appendix indicate there are decreases in pedestrian counts at these locations, there are at least four other locations where the changes are more significant but not detected by SCBS. Another location detected by SCBS that can be considered as a

false positive is Town Hall. The changes at this location are negligible and we do not show it in Figure 6.3a for brevity.

The better results of the contrast mining technique in identifying change locations is to be expected since it is a more targeted and specific approach. In addition, this experiment is also guided by the clustering result. To this end, if we use the set of locations where significant changes occur that are reported in Figure 6.3a as ground truth, SCBS correctly identifies 6 out of 10 locations. Although this is only a moderate recall, the clustering results of SCBS still indicate that the regions around which significant changes occur are correctly detected by the clustering algorithm.

6.1.3 Summary

In summary, we have applied SCBS on the data captured by the pedestrian counting system of the City of Melbourne for two different applications and produced insightful results. First, our clustering algorithm is able to detect different load profiles of pedestrians at different times of the day and at different locations in the city. We also demonstrate in the discussion that by evaluating clusters that correspond to continuous hours we can offer some preliminary insights on the flow and movement of pedestrians in the CBD. We reuse the ground truth proposed in Chapter 3 for a quantitative evaluation and conclude that the results indicate that the algorithm is able to discover different types of pedestrian distributions, and that the types of pedestrian distributions grouped by each cluster are homogeneous to some extent. We highlight that the evaluation on the grouping of dimensions into subspaces is more challenging and requires further investigation.

Nonetheless, we have demonstrated the algorithm's effectiveness in discovering subspaces in the second experiment. Here, we use SCBS to discover the impacts of a major change of public transport that was introduced in 2016. Our clustering result is then verified and is shown to be consistent with those produced by two separate techniques, which are iVAT and contrast mining. The evaluation shows that the algorithm is not only able to detect the changes in pedestrian activities but it can also detect the regions in the CBD where the changes occur.

6.2 Clustering Gene Expression Data

The development of DNA microarray technology has allowed the collection of expression levels of thousands of genes in biological experiments across different test samples or patients [104, 144]. Analyzing the patterns hidden in these gene expressions offers valuable

insights into functional genomics [114]. For example, the patterns can reveal which genes belong to the same genomic pathway that corresponds to certain symptoms of patients. However, one of the challenges of such analysis is to handle the huge amount of high dimensional data that is generated by thousands of genes. A common step toward addressing this challenge is to use cluster analysis to explore the underlying structures and interesting patterns that exist in the data, prior to further analysis.

Initial clustering algorithms for gene expression data can be classified into *gene-based clustering* and *sample-based clustering* [114, 195] algorithms. The former approach considers each gene as a point (object) and each sample as a feature of that point. This type of clustering finds clusters that contain the genes that have correlated expression levels. In contrast, sample-based clustering treats the samples (or patients) as points and the genes as features. It clusters the samples that exhibit similar levels of expressions in all the genes. However, studies [50, 114, 126] suggested that only a small subset of genes participate in the genomic pathways that correspond to a certain symptom of interest. Moreover, the symptoms of interest might only take place in a subset of samples. For this reason, it is not effective to perform clustering on the full feature space (regardless of considering genes or samples as features). This phenomenon re-iterates the challenges of finding local feature relevance as discussed in Section 2.2.1. Many co-clustering algorithms have been proposed to simultaneously cluster genes and samples of gene expression data. A discussion of existing co-clustering algorithms is presented in Chapter 2.

6.2.1 Experiments

In this section, we present an experiment to cluster gene expression data using SCBS and five other baseline algorithms. We perform clustering on ten gene expression datasets that were widely used in different studies [50, 71]. The sizes and characteristics of these datasets are summarised in Table 6.6. Here, each patient corresponds to a row, and each gene corresponds to a column of the dataset (each row is referred to as a profile of a gene). The performance of our proposed algorithm is compared with six other algorithms, including EWKM [116], BBAC [23], ITCC [70], FFCFW [214], HICC [52], and SWCC [50].

Methodology

We received the implementation of SWCC, together with the configurations to reproduce the results in [50] from the authors. We also acquired the implementation of ITCC, EWKM and BBAC. For these three algorithms, we rerun the experiments with the ten gene expression datasets using parameters as follows:

Abbr.	Name	#Patients	#Genes	#Classes
ADE	adenocarcinoma	76	9868	2
BRA	brain	42	5597	5
BR2	breast.2.class	78	4869	2
BR3	breast.3.class	96	4869	3
COL	colon	62	2000	2
LEU	leukemia	38	3051	2
LYM	lymphoma	62	4026	3
NCI	nci 60	61	5244	8
PRO	prostate	102	6033	2
SRB	srbct	63	2308	4

Table 6.6 Characteristics of 10 gene expression datasets analyzed.

- For ITCC, the number of row clusters is set to the number of distinct labels of the dataset, and the number of column clusters is set to $\{2,3,5,7,10,15,20\}$. The number of iterations is set to 30 (note that Dhillon et al. suggests that 20 iterations is sufficient for the algorithm to converge in their experiments). The coverage threshold is set to $\{10^{-3}, 5\times10^{-2}\}$.
- EWKM is also provided with the correct number of clusters, which is the number of distinct labels of each dataset. The value of λ is set to $\{0.1,0.5,0.7,1,3,5\}$. The parameter λ controls the distribution of weights of the dimensions; according to the documentation of the algorithm, these values allow for both even distribution of weights (when λ is large) and uneven distribution of weights (when λ is small). The maximum number of iterations is set to 100.
- We use *squared euclidean* as the distance in BBAC (hence the algorithm is henceforth referred to as BBAC-S). The algorithm is also provided with the correct number of row clusters. The number of column clusters is set to $\{2,3,5,7,10,15,20\}$.

Using the configurations described above, we are able to verify and replicate the results produced in [50] (based on a comparison of Normalised Mutual Information (NMI) [130] values). We are unable to obtain the implementation of HICC and FFCFW, and therefore are unable to replicate their results. We therefore quote the results produced by Chen et al. from their paper for reference.

Regarding the parameters of SCBS, in phase 1, we start the search for base clusters in two-dimensional subspaces ($2D$), and use k-means to find the base clusters in each of these subspaces. Therefore, there are only two parameters required by our algorithm: the number of base clusters k in each subspace, and the expected minimum size of a cluster, reflected

in *min_sup*. We conducted the experiment with five values of $k \in \{25, 20, 15, 10, 5\}$ and six values of *min_sup* $\in \{0.25, 0.2, 0.15, 0.1, 0.07, 0.05\}$, i.e., 30 runs in total.

Metrics

The metrics used to measure the correctness of the result are NMI, precision, recall, and F1 score. Note that *the values of precision, recall, and F1 score are not directly comparable to those results presented in the prior work* [50]. Our approach to true/false positives and true/false negatives for clustering is different from the ones used in the aforementioned paper. Specifically, after finding the clusters, these algorithms use the Hungarian algorithm [118] to find the best mapping between the clustering result and the given labels. However, the Hungarian algorithm requires that the algorithms find the correct number of clusters, which is guaranteed in [50] because this is given as an input parameter. Our algorithm does not require the number of clusters to be specified in advance, and hence it is not always guaranteed to produce the correct number of clusters. Instead, we use the approach of pair counting presented in [130] to determine true/false positives and true/false negatives. For this reason, the values of precision, recall and F1 score of the clustering results presented in Table 6.8 carry a different meaning than the results presented in [50], but we can still compare the algorithms based on the results in our experiment.

Before comparing the clustering results of the algorithms, we make the following observations on the metrics being reported. There are discrepancies between the clustering qualities reported by NMI and by precision, recall and F1 score. The discrepancies are amplified for clustering results with low NMIs and low numbers of clusters (e.g., two or three clusters). For example, although the NMI results of clustering the ADE dataset are all below 0.05, most values of precision, recall and F1 scores are above 0.5 (except for the metrics of SCBS). Looking solely as these values might lead to the false conclusion that these algorithms can find some cluster structures from the dataset. This discrepancy can first be explained by the low volumes of the datasets (the dataset with largest volume - PRO - contains only 102 data points). NMI is computed directly on the dataset, hence any wrong clustering instance directly impacts the scores. Precision, recall and F1 score use the pair counting techniques to determine true/false positives/negatives, hence the total numbers of instances used to compute the scores are in the order of $O(n^2)$. For the results with a low number of clusters, the probabilities that two points are correctly grouped together increase, which can inflate the scores of precision, recall and F1 score. We use the results of ADE to discuss this phenomenon. Although the results produced by SCBS have the highest NMI among other algorithms, the precision, recall and F1 scores are considerably lower than those produced by others. The main reason is that the baseline algorithms produce two clusters

(given by the input parameters) as opposed to 3 clusters produced by SCBS. The metrics are also arguably too high for a clustering result with NMIs in the range $[0, 0.03]$. We include the ground truth, as well as the clustering results of ADE in Table A.3 in the Appendix for reference.

Results

In this evaluation, we omit the clustering results that have NMI values near to zero since this indicates the results are as good as random assignments. These include ADE, BR2, COL, and PRO. Although the clusters produced by SCBS are better or comparable to others for these datasets, the low values of NMI suggest that such a claim is not reliable. Among the remaining six datasets, while SCBS produces better results than ITCC, FFCFW and HICC on most of the datasets, it is noticeably outperformed by BBAC and SWCC on datasets BRA, BR3, LEU, LYM and NCI.

The poor clustering results produced by SCBS can be accounted for by the small volumes, and the imbalance between the numbers of rows and numbers of columns (the maximum number of rows is 102 while the minimum number of columns is 2000). This results in short but very wide transactional tables after the first phase of SCBS, on top of which the FP Trees are constructed in the second phase. Note that the output transactional table Z of phase 1 has the size of $n \times \frac{d(d-1)}{2}$ for a $n \times d$ dataset (without subspace sampling). These short but wide transactional databases raise challenges in constructing frequent patterns, mainly due to low support that can make the frequent patterns more susceptible to noise created from an incorrect clustering. In fact, the dataset LEU has only 38 data points and each data point corresponds to a support of 2.6%.

We also assess the number of clusters being produced by SCBS. Note that other algorithms are provided with the correct number of clusters as an input parameter, hence this metric is not presented in Table 6.7. The average number of clusters produced by SCBS on ADE, BRA, LYM and PRO are fairly close to the number of labels in the ground truth. Our algorithm SCBS also produces large numbers of clusters (> 10) on four datasets. These results correspond to the parameters with low values of `min_sup` ($min\_sup < 0.1$). Such a parameter setting results in multiple frequent patterns that match to different small subsets of points and results in a high number of clusters compared to the number of labels in the ground truth. In addition, since SCBS does not exhaustively assign every point to a cluster (points that do not have their items match to any frequent pattern are outliers), we also analyze the average coverage of clusters. Table 6.7 shows that the coverage is only moderate, with the highest of 65%, and lowest of 44%. We link this phenomenon of low coverage to the low volumes of the inputs. Here, in the second phase of the algorithm, less than 100 transactions

Table 6.7 Statistics of the number of clusters and coverage of points being clustered by SCBS.

Datasets	ADE	BRA	BR2	BR3	COL	LEU	LYM	NCI	PRO	SRB
Min number of clusters	2	1	1	1	2	1	1	1	2	2
Max number of clusters	4	13	11	10	9	7	7	12	9	8
Average number of clusters	2.2	6.0	6.8	6.3	4.4	3.8	2.8	4.9	2.1	2.8
Coverage of clustered points	65%	53%	49%	50.1%	58%	57%	48%	63%	51%	44%

provide support to find the frequent patterns for millions of items (the number of $2D$ clusters). In addition to the unreliability of the process, the discovered frequent patterns are usually long and do not easily match to the items of the points.

In summary, we have explored the ability of SCBS to act as a co-clustering algorithm in this experiment to cluster high dimensional gene expression data. While we observe that in some cases the cluster structure can be found and the results are better than some popular co-clustering algorithms (i.e., ITCC, FFCFW and HICC), SCBS tends to be outperformed by the more recently proposed algorithms, i.e., BBAC and SWCC. This experiment also reveals a limitation of SCBS that does not allow it to work with data having low volume but high dimensionality. As the transactional database used for frequent pattern mining in the second phase has n rows, each having $O(d^2)$ items, it is essential for n to be relatively large to support the frequent pattern mining process and make it more tolerant to any noise created by false clustering instances.

In contrast, the co-clustering algorithms consider the rows and columns interchangeably and are designed to work with datasets that have imbalanced cluster sizes. Therefore, we conclude that SCBS, in its current state, is not suitable to be used with that type of data such as gene expression micro array datasets.

6.3 Using Clustering in an Ensemble Classification Model

We demonstrate the capability of our algorithm to work with data in another IoT application. In this experiment, we explore the ability of SCBS to produce clustering results that can be used as an intermediate step to assist the construction of a more complicated model.

Specifically, we use the clustering result to build an ensemble classification model and show that the model improves the accuracy in predicting the car parking occupancy in the CBD of Melbourne.

Table 6.8 Comparison of the clustering results (using NMI, precision, recall and F1 score) of SCBS with six other clustering algorithms.

Data	Metrics	SCBS	EWKM	BBAC-S	ITCC	SWCC	FFCFW	HICC
ADE	NMI	0.03	0.00	0.00	0.01	0.02	0.00	0.01
	Precision	0.74	0.73	0.73	0.73	0.75	-	-
	Recall	0.34	0.50	0.50	0.5	0.53	-	-
	F1 Score	0.47	0.60	0.60	0.60	0.62	-	-
BRA	NMI	0.25	0.19	0.45	0.00	0.40	0.00	0.13
	Precision	0.35	0.31	0.53	0.22	0.53	-	-
	Recall	0.27	0.30	0.46	0.22	0.46	-	-
	F1 Score	0.30	0.31	0.50	0.22	0.49	-	-
BR2	NMI	0.05	0.01	0.04	0.00	0.06	0.00	0.01
	Precision	0.55	0.51	0.53	0.52	0.55	-	-
	Recall	0.28	0.50	0.52	0.53	0.54	-	-
	F1 Score	0.37	0.51	0.53	0.53	0.54	-	-
BR3	NMI	0.09	0.08	0.21	0.00	0.22	0.00	0.02
	Precision	0.4	0.41	0.50	0.38	0.48	-	-
	Recall	0.38	0.38	0.37	0.35	0.39	-	-
	F1 Score	0.39	0.40	0.43	0.36	0.43	-	-
COL	NMI	0.07	0.02	0.04	0.00	0.04	0.00	0.01
	Precision	0.60	0.56	0.58	0.55	0.57	-	-
	Recall	0.25	0.52	0.56	0.51	0.52	-	-
	F1 Score	0.35	0.54	0.57	0.53	0.55	-	-
LEU	NMI	0.13	0.21	0.43	0.00	0.34	0.00	0.05
	Precision	0.69	0.73	0.84	0.58	0.77	-	-
	Recall	0.34	0.53	0.54	0.54	0.59	-	-
	F1 Score	0.46	0.61	0.66	0.56	0.67	-	-
LYM	NMI	0.2	0.33	0.62	0.00	0.47	0.07	0.04
	Precision	0.78	0.84	0.94	0.51	0.88	-	-
	Recall	0.41	0.58	0.80	0.34	0.58	-	-
	F1 Score	0.53	0.69	0.86	0.41	0.70	-	-
NCI	NMI	0.3	0.23	0.61	0.00	0.53	0.24	0.15
	Precision	0.28	0.22	0.60	0.21	0.87	-	-
	Recall	0.33	0.23	0.53	0.23	0.59	-	-
	F1 Score	0.30	0.22	0.56	0.22	0.70	-	-
PRO	NMI	0.05	0.01	0.02	0.00	0.03	0.00	0.01
	Precision	0.53	0.51	0.51	0.50	0.51	-	-
	Recall	0.53	0.51	0.52	0.51	0.53	-	-
	F1 Score	0.53	0.51	0.52	0.51	0.53	-	-
SRB	NMI	0.25	0.14	0.26	0.00	0.20	0.00	0.07
	Precision	0.57	0.39	0.55	0.29	0.46	-	-
	Recall	0.31	0.35	0.41	0.27	0.34	-	-
	F1 Score	0.40	0.37	0.47	0.28	0.41	-	-

6.3.1 Related Work

This approach of using clustering in an ensemble prediction model has previously been used in [28, 41, 80, 215, 216].

Benmouiza et al. [28] use k-means and artificial neural networks [57] to build an ensemble prediction model to forecast hourly global horizontal solar radiation. First, k-means clusters the time series data into groups that possess similar patterns of solar radiation, which are called forecasted regions. In the second phase, a multi-layer perceptron neural network [87] is applied on each forecasted region to perform multi-step ahead prediction of the hourly solar radiation. The study shows that clustering not only helps interpret the behaviour of the time series data, but also provides better forecasting results by learning different subsets of data using different submodels.

Trivedi et al. [215] study the use of predictors in conjunction with clustering algorithms. They first use k-means to produce k clusters. A predictor is then trained on each cluster, resulting in k predictors, which are subsequently combined by averaging the regressed values. The experiment is performed on different predictors, including Linear Regression, Step-Wise Linear Regression [194] and Random Forests [137], and evaluated on ten different datasets [69]. The experiments demonstrate that the ensemble model improves the prediction accuracy in most datasets.

Fahiman et al. [80] propose a new approach to the problem of short term forecasting of electricity usage from smart meter data of the Commission for Energy Regulation [10] using a combination of K-shape clustering [165] and deep learning [27, 131]. Specifically, the first phase of the algorithm uses K-shape similarity to cluster the electricity usage data into clusters of households that share the same usage profile. A deep learning model is applied on each cluster and the final prediction result is combined via a weighted aggregation mechanism. The experiments show that this proposed method produces the best accuracy when compared with other prediction methods in the literature [100].

These studies focus on the second phase of building the ensemble models, whereas the first phase usually employs quite simple techniques such as k-means. We believe this might underestimate the complexity of the structure of the datasets in real-life applications. In fact, if the clustering algorithm does not group similar training examples to the same clusters, the submodels that are built on these clusters can potentially have lower quality, which subsequently leads to lower performance of the prediction model, despite the cost of extra computation. We present this phenomenon in this experiment. We demonstrate that our subspace clustering algorithm can learn the structure of a complicated real-life dataset, which subsequently leads to a better accuracy of the ensemble classification model.

6.3.2 Experiments

The City of Melbourne has deployed sensors to record parking events at parking bays around the central business district (CBD). The data is available on the open data portal of the City of Melbourne [150]. We extract the start and end time of all parking events to compile the parking occupancy at 276 locations at 15 minutes intervals between 09:00-18:00, yielding an input of size 276×36 for each day. Parking occupancy is an important metric that indicates the efficiency of car park utilisation [123], which heavily affects traffic, ease of commute and business in the CBD. Analysing the car occupancy can reveal patterns in parking behaviour at different car parks during different times of the day, which can then be used to review the parking hotspots or tariffs.

We formally state the problem next. Let X be a collection of N car parking occupancy values $X = \{x_i : i = 1..N\}$, where $x_i = \{x_{i_1}, x_{i_2}, ..., x_{i_j} ..., x_{i_d}\}$ consists of the car parking occupancy at location i at different times $\{t_j\}_{j=1}^{d}$ of the day. Here, each row corresponds to a location and each column corresponds to an interval between 9:00 and 18:00. In this study, we use SCBS on the car parking occupancy datasets in order to *find clusters of car parking spots that have similar patterns of occupancy at certain times of the day*. We then use this clustering result to construct an ensemble prediction model to predict future values of car parking occupancy at all locations using historical values. Specifically, the models use $\{x_{i_j}\}_{j=1}^{d-\Delta}$ *to predict the subsequent* Δ *values of car parking occupancy for location* i. Our hypothesis is that if the parking spots being grouped to the same cluster indeed exhibit the same patterns of parking occupancy, it will be easier to learn the patterns and predict the parking occupancy by looking into each cluster separately. To this end, an ensemble regression model that contains multiple submodels, each of which is built on data collected at a cluster of locations, *is expected to have better performance than a single regression model built on the entire dataset.*

The evaluation of the prediction model serves two purposes:

- First, it can be used to quantify the effectiveness of the clustering algorithm. Any improvement of the predicted results can be directly accounted to the quality of discovered clusters that contain locations that exhibit similar patterns of parking occupancy. In addition, while we have access to the parking datasets, we are unable to acquire the mapping of sensor ID to the actual location of the parking spots in the CBD, which limits our options to discuss the results. To this end, this method provides us with a means to verify the clustering results.

- Second, we aim to use the improvement in the performance of the prediction task as an example to illustrate that the application of SCBS can be widened beyond the scope of

clustering. In this case, we aim to show that SCBS can be used with high dimensional data as an intermediate step to assist more complicated decision making processes, such as prediction in an IoT application.

Each clustering task is performed on five days worth of data (all weekdays). As presented above, the data collected for each day has a size of 276×36, resulting in sub-datasets with the sizes of 1380×36 being clustered. Subsequently, these datasets are used to train the ensemble model to predict car parking occupancies in the following hours. More details of the experiment designs, implementation, baselines as well as evaluations of the model are discussed in the next sections.

With the intention of using the clustering results to build an ensemble prediction model, we indirectly control the number of clusters, i.e., the number of different groups of locations, in this experiment. Specifically, we start the search for base clusters in $2D$ base subspaces, using k-means with $k \in \{2,5,10\}$ and $min\_sup \in \{0.05,0.1,0.15,0.2\}$. The first clustering result that has five clusters is selected. Note that this number is decided somewhat arbitrarily. The number of clusters is also the number of sub-models being built in the ensemble model. A high number of clusters might lead to the ensemble model not able to generalize and hence overfitting. In contrast, a low number of clusters might not result in a noticeable improvement of the prediction performance compared to a single prediction model.

Clustering Car Parking Occupancy Data

By clustering the parking occupancy data, each cluster $C_{S_i}^{X_j}$ represents a parking pattern observed at the locations (points) X_j during the times (dimensions) defined by S_i. Here, each parking sensor deployment location is only represented by an ID and we are unable to acquire the actual locations of the sensors to conduct a qualitative evaluation of the grouping of locations. Instead, we discuss some potential qualitative evaluation methods if such information becomes available.

- The locations can be classified into areas of offices, restaurants, shopping malls, etc. and the evaluation can analyze the location types of each cluster. In addition, some simple verification can be done to further assess the subspaces of clusters. For example, simple rules can be proposed as ground truth, such as "clusters that mostly contain parking spots in restaurant areas are formed in dimensions that correspond to lunchtime and dinner hours."

- In a less granular evaluation, the locations can also be categorized into buckets depending on their distances to the centre of the CBD. These buckets can be used as ground

truth. The hypothesis this ground truth can verify is that the car parking occupancy is consistently higher in the centre of the CBD (and hence clustered to the same group), and decreases as the distance to the CBD increases.

To this end, we perform an internal cluster evaluation by analyzing the cluster coherence, in addition to the attributes of cluster sizes and cluster subspaces. The evaluation of the ensemble prediction model serves as an external evaluation. Table 6.9 summarizes the results of 50 instances of clustering that we perform in this experiment (each clustering task is done on a subset of five weekdays' of data, i.e., a 1380×36 dataset). The smallest cluster size has 133 points and the largest cluster has 347 data points. Note that each data point corresponds to a parking spot, and each parking spot has a frequency of 5 in each dataset (one for each day). Since the clustering result produces the grouping of parking spots to segment each dataset into subsets to train the submodels, it is important to analyze the number of unique parking spots of each cluster. This is to ensure that we do not have significantly unbalanced groups. Table 6.9 shows that the smallest group contains 68 parking locations and the largest group contains 157 locations. In addition, the average cluster size and moderately low standard deviation values show that it is unlikely that the training data for submodels are highly unbalanced. More details on how the datasets are segmented to subsets for submodel training are given in the next section.

Table 6.9 Statistics of cluster sizes and subspaces of 50 clustering instances.

Metrics	Values
Minimum cluster size	133
Maximum cluster size	347
Average and standard deviation of cluster size	230 ± 46
Minimum count distinct locations in one cluster	68
Maximum count distinct locations in one cluster	157
Average and standard deviation of distinct locations	110 ± 21
Minimum cluster dimensionality	5
Maximum cluster dimensionality	21
Average and standard deviation cluster dimensionality	13.2 ± 5.2

We analyze the coherence of each cluster by statistically verifying whether the clustered parking bays have small deviations in the values of parking occupancy during the corresponding time periods, compared to the rest of the data. The examples of two clusters are shown in Figure 6.5, where each blue bar represents the mean and standard deviation of the parking occupancy at a certain time of the day, observed at parking bays grouped by the cluster. For example, `Cluster 1` in Figure 6.5a shows the pattern shared by a group of parking bays during 9:00-10:30 and 14:45-17:45 with small standard deviations, compared to significant

deviations at other times of the day. Similarly, `Cluster 2` shows another pattern that has an occupancy rate of 55% around midday, while such correlation is not observed at other times of the day.

Ensemble Regression Model

Next we use the clustering result to construct an ensemble prediction model to predict parking occupancy in the afternoon using the values observed in the morning of the same day. Each cluster ideally represents a pattern of parking occupancy shared by a group of parking bays. Fitting a submodel to each cluster allows each submodel to learn the data in more detail and predict with higher accuracy if the values are coherent. Therefore, the accuracy of the prediction model directly reflects the quality of the clusters.

The construction of the ensemble model follow these steps:

- **Step 1: Refine the locations in each cluster**. We perform clustering on each subset of five days' worth of data. Therefore, each location can be present up to five times in a cluster. This step removes duplicates and further refines the locations in each cluster. Specifically, any location that is present more than three times is retained, and any location with lower frequency is removed. All the locations that are either removed or not covered by any cluster are added to a new cluster. With the design of this experiment, we have six clusters of locations (five are generated by the algorithm and the sixth cluster contains locations that do not belong to any cluster) that are used to build the ensemble model.

- **Step 2: Segment the training dataset**. The data that is used for training is further segmented into six subsets, where each subset contains only the observations collected at locations that belong to a cluster. Note that prior to the segmentation of the training set, we use a stratified sampling strategy to split the data into training and test sets with a ratio of 80% training and 20% test data.

- **Step 3: Train a submodel on each subset of data**. We use the well-known `scikit-learn`[1] library to train a simple decision tree regression model [235] on each subset of the data, using 10-fold cross validation to prevent overfitting. The input variables are the parking occupancy values captured at 15 intervals from 09:00 to 12:45 and the output variables are eight values of the car parking occupancies to be predicted from 13:00 to 15:00.

[1] https://scikit-learn.org/

- **Step 4: Ensemble the submodels**. The final model is the ensemble of all six submodels, and the task of predicting the car occupancy at a certain location is delegated to the submodel that is trained on the observations collected at the relevant location groups. To this end, this step implements a simple wrapper that routes an observation that needs to be predicted to the relevant submodel.

We propose two baseline algorithms for comparison. First, in order to claim that the approach of building the ensemble model can improve the prediction performance, we propose the first baseline as a single decision tree regression model that is trained in one pass with no segmentation of locations. Second, we also aim to show that SCBS can produce better clustering results on this dataset. Hence, we construct another ensemble model using the same steps described above, except that the clustering result in **step 1** is produced by k-means (with $k = 6$) instead of SCBS. In summary, we compare the performance of three prediction models as follows.

- Model 1 applies decision tree regression directly on the occupancy data with no grouping of locations.
- Model 2 first clusters the data using SCBS and then fits a decision tree regression on the data collected at car parks in each cluster separately.
- Model 3 follows the same approach as Model 2 except that it uses the k-means algorithm in the first phase.

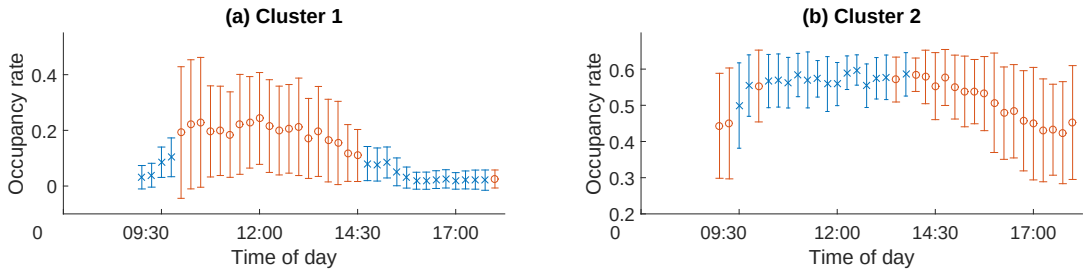


Fig. 6.5 Illustration of cluster quality.

In each evaluation instance, each model is trained and evaluated on five weekdays' worth of data (the split of training and test set is 80%:20%). We perform 50 evaluation instances on the data collected during 50 weeks of 2014 (the first week in January and last week in December are excluded since the traffic in the CBD during these weeks is known to be significantly different). The coefficient of determination (R2) [164] is used to measure the performance of the models on the test sets. The results are summarized in Table 6.10 and Figure 6.6.

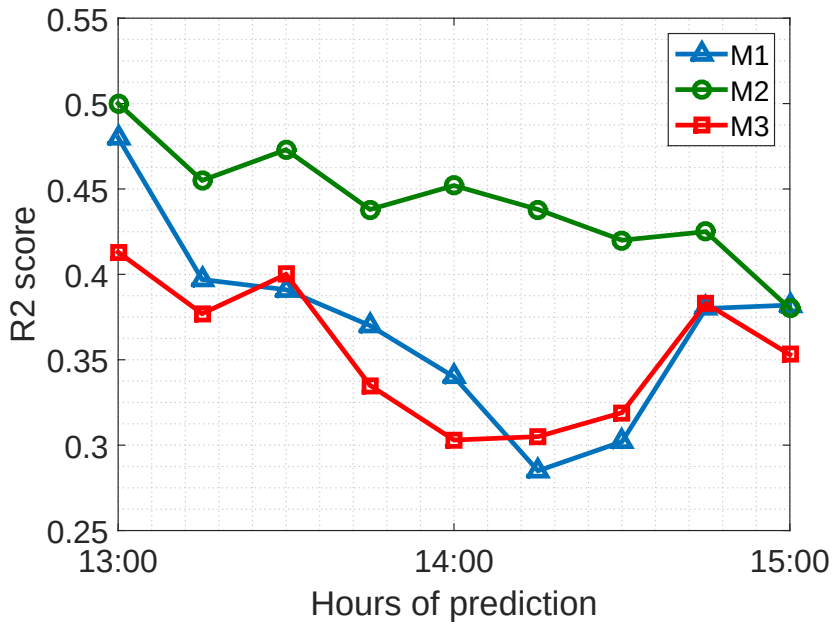


Fig. 6.6 Prediction accuracy in terms of R2 score of three models: M1 - decision tree regression on all data, M2 - separate decision tree regression on each cluster using our algorithm, M3 - separate decision tree regression on each cluster using k-means.

Table 6.10 Prediction accuracy in terms of R2 scores of the three models for predicting car parking occupancy at different hours from 13:00 to 15:00.

	13:00	13:15	13:30	13:45	14:00	14:15	14:30	14:45	15:00
M1 (single model)	0.48	0.40	0.39	0.37	0.34	0.29	0.30	0.38	0.38
M2 (ensemble model using SCBS)	0.50	0.46	0.47	0.44	0.45	0.44	0.42	0.43	0.38
M3 (ensemble model using k-means)	0.41	0.38	0.4	0.34	0.30	0.31	0.32	0.38	0.35

In summary, we explore and widen the applications of SCBS to a prediction task using a high dimensional and complex data generated by an IoT application. Similar to other studies on the same topic, we demonstrate that the extra computation spent on clustering and building the ensemble model can be justified by the improved performance of the prediction model. We also go further than other studies by showing that the effect of the clustering results on the construction of the ensemble model is indeed critical, and that the observed improvement in the prediction model is not always guaranteed, as we show when k-means is used in this experiment. Using simple k-means and the traditional regression model as baselines, we

demonstrate the potential value in adopting SCBS as an intermediate step to enhance more complicated decision making processes, especially when high dimensional data is involved. Nevertheless, the usefulness of SCBS is demonstrated with respect to fairly simple baselines. Therefore, as a future research direction, more experiments can be conducted to evaluate the effectiveness (how much the prediction performance is improved) and efficiency (how much more complexity is introduced) of using SCBS compared to using other high dimensional data clustering algorithms for this task.

6.4 Summary

In this chapter we presented the use of SCBS to cluster multi-dimensional and high dimensional data generated by three different real-life applications. All the datasets being analyzed vary in volume, number of dimensions, as well as complexity. The experiments help to not only demonstrate the effectiveness but also reveal some limitations of our proposed algorithm.

In the first application on pedestrian counting data, we show that SCBS can discover clusters that can be mapped to different types of distributions of pedestrians at different periods of the day with moderately high values of precision, recall and F1 score. Although we believe more complicated and more granular distributions of pedestrians exist, the clustering results we achieve indicate that SCBS is able to reveal these clusters to provide interesting insights. We also provide a preliminary example on how the interpretation of clusters that correspond to continuous hours can provide an overview on the flow of pedestrians in the CBD. We then use SCBS to analyze the impacts of the Night Network trial, which is a major change in the transport system, on the activities of pedestrians during midnights on weekends. Specifically, the changes of pedestrian distributions at 1am and 2am of Saturdays and Sundays are reflected as two separate clusters that SCBS is able to discover. The verification with iVAT and a contrast mining technique validate the clustering results, both on the changes of pedestrian distributions and on the locations where the changes take place.

In the second application, we explore the feasibility of using SCBS in a new domain, which is to cluster the high dimensional data of gene expressions. To this end, the algorithm is evaluated with six other co-clustering algorithms on ten popular gene expression datasets. One of the insights gained from this experiment is that clustering gene expression data remains a challenging task, and all algorithms are not able to find clusters that are better than random assignments on four datasets. Among the remaining six, although SCBS produces better results than three algorithms, it is significantly outperformed by more recent algorithms. This experiment also highlights a limitation of our algorithm. As we push the algorithm to

handle higher dimensional data, we overlook a requirement that SCBS needs the input to have large volumes relative to the number of dimensions to provide sufficient support for the frequent patterns in the frequent pattern mining process. This requirement becomes even more essential for high dimensional data as the number of items ($2D$ base clusters) generated grows quadratically w.r.t the number of dimensions. We conclude that while SCBS can find clusters in some datasets, it is in general not as effective as other state-of-the-art co-clustering algorithms. Hence, addressing low volume high dimensional data can be a future research challenge that needs to be addressed to make SCBS's performance more comparable to those of co-clustering algorithms for such datasets.

We apply SCBS on another real-world dataset generated by an IoT application and aim to show that the results can be used outside the scope of a clustering application. Although the method of building an ensemble classification on clustering results has been applied in different applications before, we show in this experiment that a complex and multi-dimensional dataset needs an adequate algorithm to find good quality clusters for the ensemble model, otherwise the overall performance can deteriorate despite the extra computational cost. Due to the lack of ground truth for evaluation, the better prediction performance of our ensemble model compared to the baselines can justify the clustering qualities. The results suggest that SCBS can be used as an intermediate step in other machine learning tasks when high dimensional data is involved.

By showing the applications of SCBS on multiple real-world datasets that have different volumes, numbers of dimensions, and levels of complexity, we demonstrate the potential in using SCBS in practical applications to produce interesting and useful insights.

Chapter 7

Conclusion and Future Research

7.1 Summary of Contributions

We began our research by performing experiments on the data collected by an IoT application to gain a better understanding on the topic and the properties of real-life data. Specifically, we aim to analyze different profiles of pedestrian distributions using the pedestrian count data collected by the City of Melbourne, and subsequently use the profile to detect anomalies. In Chapter 3, we use clustering and frequent pattern mining to build two different models of the pedestrian distribution, and extract different profiles of distributions that correspond to pedestrian activities in different locations in the City of Melbourne. We also use the models to identify anomalies in pedestrian activities during holidays, festivals and sporting events. Although the experiments produce positive results, they are conducted only on low dimensional data. Specifically, despite the use of categorization to decrease the complexity of the data, we are unable to apply the same approach on data with more than 10 locations. These experiments motivate our subsequent research on subspace clustering and allow us to identify the main challenges in handling higher dimensional data. These challenges include (1) to find the relevance of features in different local subspaces in order to effectively measure similarities between points, (2) to find clusters in non-disjoint subspaces, and (3) to scale the measurements and the clustering algorithm to data with large volumes and high dimensionality. We subsequently address these research challenges in the following chapters.

We propose the local similarity measure in Chapter 4 that can find locally relevant subspaces to provide an effective measure that can reflect the similarities between data points in high dimensional spaces. The measure assesses the similarities in 2D base subspaces before identifying the subspaces in which the dimensions are relevant. Subsequently, the similarity is computed only in that relevant subspace. We provide several proofs of concept for similarities based on distance, density and grids. This enables our proposed similarity

measure to be used with different clustering algorithms. Our evaluation shows that this proposed method can overcome the problem of local feature relevance and can find the relevant subspace before effectively computing the similarity between two data points in high dimensional space. We also discuss the benefits of searching for base similarities in 2D subspaces, and how it can help preserve the covariances between dimensions, despite the extra complexity it introduces to the algorithm. We then analyze the time complexity and propose an improved version of the algorithm that uses subspace sampling, which reduces the time complexity while still ensuring *equal presence of every dimension*. The similarity measure, while being effective in measuring point-to-point similarity, cannot be used as it is in an efficient clustering algorithm. First, the search for locally relevant subspaces and the subsequent computation of similarities for *all pairs of points* are computationally expensive and significantly limits the algorithm's scalability. Second, the similarity measure is subspace-context based, i.e., any grouping of points into clusters also needs to take into account both similarities and the accompanying subspaces. We address these challenges in Chapter 5 and propose our subspace clustering algorithm.

In Chapter 5, we integrate our proposed similarity measure into our proposed algorithm for Subspace Clustering by aggregation of Base Similarities (SCBS) that can scale to large volumes of high dimensional data. Our proposed clustering algorithm consists of two phases. Supported by our similarity measurement, it first searches for base clusters in low dimensional subspaces before aggregating them to form clusters in higher dimensional space in the second phase. The use of any available clustering algorithms to find base clusters in the first phase is possible, e.g., density-based, distance-based or grid-based clustering, giving considerable flexibility to our proposed method. The second phase also reflects the novelty of our algorithm. Specifically, we propose to use frequent pattern mining to combine base clusters together, instead of the sequential bottom-up approaches of existing algorithms. This avoids the combinatorial complexity of those algorithms and allows our algorithm to scale to data with high dimensionality. This approach also allows the algorithm to avoid the construction of a similarity matrix, which reduces both time and space complexity and enables the algorithm to work with inputs that have very large volumes. The aggregation also allows a base cluster to be part of multiple higher dimensional clusters, which enables the algorithm to find clusters in non-disjoint subspaces. We then analyze its time complexity and propose a subspace sampling technique to be incorporated in the first phase of the algorithm. This allows our algorithm to cluster a reasonably large dataset that has 10,000 data points and 3,500 dimensions. We thoroughly evaluate the algorithm with multiple synthetic datasets having different sizes, dimensionalities and level of outliers, in both disjoint and non-disjoint subspaces, and demonstrate that it produces comparable or better results than state-of-the-art

algorithms. We also show that it can cluster up to 1 million data points in reasonable time and is faster than other algorithms.

In Chapter 6, we proceed to demonstrate the practicality of our proposed algorithm with real-life applications in different areas. First, we use the proposed clustering algorithm to find different profiles of pedestrian distribution across the CBD of Melbourne. The clustering results are evaluated both qualitatively and quantitatively, and we show that they offer intuitive insights into the distribution and movements of pedestrians in the city. This application shows that our subspace clustering algorithm is able to achieve good clustering results on more complete and higher dimensional data that we were unable to cluster in Chapter 3. We then use the clustering algorithm to discover the difference in pedestrian distribution over a major change of the public transport system of Melbourne. The results are evaluated and validated with two other techniques of iVAT and contrast mining. We show that the two contrasting distributions discovered by SCBS are valid, and that the regions where the changes occur are identified with high accuracy by the clustering algorithm. Second, we explore the capabilities of our algorithm in clustering data generated by bioinformatics applications, in comparison with other co-clustering algorithms. We evaluate the performance of the algorithms on ten datasets of gene expression measurements to simultaneously group genes and patients that express the same patterns of expression levels into clusters. The results show that although SCBS produces better results than three algorithms, it is outperformed by more recent algorithms. The results suggest that SCBS is not as effective as other state-of-the-art co-clustering algorithms for the task of clustering highly sparse data that has high dimensionalities but very low volumes. We identify this as a limitation and a potential future research direction to improve our subspace clustering algorithm. In the next application, we use the clustering results produced by our algorithm to build an ensemble classification model and use it to learn and predict the car parking occupancy at 276 locations in the City of Melbourne. The empirical results show that the ensemble model improves the performance of prediction. This demonstrates the potential of using our subspace clustering algorithm as an intermediate step in more complicated machine learning tasks to analyze high dimensional data.

In summary, our proposed methods offer novel approaches to measure similarity and cluster data in high dimensional space. The evaluation with synthetic and real-life data shows that it is able to find subspace clusters in non-disjoint subspaces with high accuracy, to handle data with complex structures and high levels of outliers, and to scale to large volumes of data with high dimensionality. These attributes make the algorithm practical for many applications in real life.

7.2 Future Research

We now discuss several future research directions, which are motivated by this thesis and are open challenges for subspace clustering of high dimensional data.

Clusters in Arbitrarily Oriented Subspaces

Our focus in this thesis is to find subspace clusters in axis-parallel subspaces. This class of subspace clustering algorithms is useful in applications where the value in each dimension is considered as a feature, and has a real meaning to users. These applications include bioinformatics, sensor data and financial data. Another class of subspace clustering algorithms aims to find clusters in arbitrarily oriented subspaces. These algorithms are useful for applications such as computer vision, image and video processing. For example, given face images of multiple subjects, the images that belong to the same subject ideally should be clustered into the same cluster.

Our proposed method can be extended to find base clusters in arbitrarily oriented $2D$ subspaces. This can be achieved by performing rotation on the two axes and projecting the data on the rotated coordinates before performing clustering. Studies [128] have pointed out that if data are correlated, they are closely distributed when projected on a certain hyperplane. The base clusters are no longer parallel to the axes and the aggregated clusters are therefore arbitrarily oriented. More research needs to be done to verify that the aggregation of base clusters form a cluster in high dimensional space. In addition, although the suggested method of rotating axes provides an intuitive way to find arbitrarily oriented base clusters, it is limited to only $2D$ subspaces and can be relatively expensive due to the task of rotating and clustering. An alternative solution is to investigate the use of *correlation clustering* [15, 208] to find arbitrarily oriented base clusters in low dimensional space. Moreover, in order to analyze the relationships between clusters in the final results, it is necessary to develop a mechanism to identify the orientation and the dimension components of each cluster, which is trivial in axis-parallel subspaces.

Sampling Mechanism of Base Subspaces

Our proposed method searches for base clusters in all p-dimensional subspaces. This leads to quadratic computational complexity (for $p = 2$) or even combinatorial complexity as p increases. In this thesis, we address this problem by proposing to randomly sample the dimensions in each iteration for the base subspaces in which the base clusters are formed. This approach is simple and is shown to work effectively in Section 6.2.

Since this process directly affects the clustering quality and efficiency of the algorithm, further research can investigate other methods of sampling or different mechanisms to initialize base subspaces. A possible method is to have a more controlled sampling procedure to closely monitor the involvement of each dimension into base clusters. This approach, together with Lemma 4.3.2, allows the algorithm to compute the radius of the final clusters.

In addition, if some knowledge of dimensions (features) of the data is known in advance, it can be advantageous to apply stratified sampling [241] on the dimensions. For example, in a dataset collected by 200 sensors (200-dimensional dataset) in 30 locations, data collected by nearby sensors are strongly correlated. Therefore, if the task is to explore patterns in the sensor data at different locations, searching for base clusters in the subspaces formed by nearby sensors might not yield any new interesting information. To this end, a stratified sampling method can be applied, where 200 dimensions can be divided into 30 groups (or strata [241]), each corresponding to the sensors at a location. This approach prevents unnecessary searching of base clusters (i.e., in subspaces formed by sensors at the same location) and guarantees that each base cluster expresses correlations at different locations.

Another potential research direction is to have a sampling procedure that progresses and evolves with the search of base clusters. The existing base clusters can provide an early indication of the likelihood of cluster formation in certain subspaces. Therefore, it is possible to use this information to disregard at an early stage the subspaces that do not participate in any high dimensional clusters to further improve efficiency.

Improved Aggregation of Base Clusters

The aggregation phase of our algorithm mines only maximal frequent patterns. This approach allows removal of redundant clusters and keeping the number of clusters tractable, but may also remove some interesting clusters. More sophisticated measures can be applied to aggregate and extract clusters. The next step can compute the overlap between clusters and have an additional step to either merge overlapping clusters or prune redundant ones. This leads to several research challenges. The first is to maintain the low computational complexity of the algorithm given the additional requirements of mining more frequent patterns and computing cluster overlap. The second challenge is about computing the overlap of clusters, taking into consideration both points and dimensions simultaneously. In addition, the new method also needs to avoid the formation of large but non-meaningful clusters (due to excessive merging).

Another potential solution that can be further investigated is to formulate the aggregation of base clusters as a combination of frequent pattern mining and set cover [201]. To this end, each frequent pattern being mined is considered as a subset, and the extraction of

clusters becomes a problem of finding a set of frequent patterns that best cover the points and dimensions of the data. Here, the problem is more challenging because the number of final sets is not known (in contrast to a pure set cover problem). In addition to solving the set cover problem, the algorithm also needs to find a balanced number k of sets, i.e., the number of clusters. If k is set too high, the algorithm might result in multiple small clusters that do not express interesting patterns. On the other hand, if k is not sufficiently high, the aggregation might result in only large clusters and overlook smaller interesting patterns.

Subspace Clustering in Dynamic Systems

The proposed method and the applications so far assume the systems in which the data is collected are static. It means any patterns found by the clustering algorithm do not change over time. This assumption can be impractical, especially when analyzing data from dynamic environments over a prolonged period. In fact, the advantages of considering dynamic states of the system have been presented and discussed in Section 3.3.

A direction for future research is to detect different states that exist in the dynamic system from which the data is collected. This raises multiple research challenges in high dimensional space. The changes can be reflected in either the samples (points), or correlations between dimensions, or both. Therefore, it is necessary to construct a model to characterize each state of the system; the model needs to be able to facilitate the comparison between states, in order to detect the switching of states in the system. We believe this feature of the algorithm can provide useful insights into how the subspace clusters evolve over time, and allow the algorithm to study dynamic systems that switch intermittently between states.

Locality Sensitive Hashing in Subspace Clustering

Locality sensitive hashing (LSH) is a well-known technique to handle high dimensional data [59, 142]. LSH hashes input items with the intention of maximizing the probability of collisions for similar items, in order to put these items into the same bucket. LSH has been used widely in the applications of nearest neighbor search [106, 109, 191]. However, it has limited applications in clustering, especially clustering high dimensional data [54, 95, 127]. Nevertheless, in these applications, LSH is only involved in one of the steps of the clustering process. For example, Hisashi et al. [127] use LSH only to initialize the initial state before performing optimization to achieve clustering.

We believe that LSH has the capability of computing similarity and subsequently can be used more in applications of clustering high dimensional data. In particular, by grouping similar points into the same buckets, LSH offers some degree of similarity between points.

However, LSH includes all dimensions when computing hashes, therefore it does not account for local feature relevance between different subsets of points. This is one of the main research challenges that needs to be addressed. Inspired by our similarity measure, a potential solution is to perform LSH on sampled subsets of dimensions in multiple iterations, before aggregating the hash results to derive the final similarity. More research needs to be done on the sampling mechanism, the aggregation technique of hash buckets, as well as the feasibility of such an approach.

References

[1] City of Melbourne : Pedestrian Counting System. http://www.pedestrian.melbourne.vic.gov.au/.

[2] Events Calendar for 2014 - Victorian Government. http://www.vic.gov.au/calendar/2014/01.html.

[3] Internet of Things (IoT) - ISSNIP. http://issnip.unimelb.edu.au/research_program/Internet_of_Things.

[4] Night Network Overview - Public Transport Victoria. https://www.ptv.vic.gov.au/getting-around/night-network/night-network-overview/. Accessed: 2018-08-26.

[5] Pedestrian Count Study - Downtown Raleigh Alliance. https://www.yumpu.com/en/document/view/21531567/pedestrian-count-study-downtown-raleigh-alliance.

[6] Pedestrian counters in the City of Melbourne. https://wongm.com/2012/10/city-of-melbourne-pedestrian-counters/.

[7] San Francisco parking data. http://sfpark.org.

[8] SmartSantander. http://www.smartsantander.eu.

[9] What's On in Melbourne December 2014. http://www.victoria.aussietrueblue.com/what-to-do-in-Melbourne-December-2014.php.

[10] Smart Metering Trial Data Publication. http://www.cer.ie/en/information-centre-reports-and-publications.aspx?article=5dd4bce4-ebd8-475e-b78d-da24e4ff7339, 2018.

[11] P. K. Agarwal and N. H. Mustafa. K-means projective clustering. In *Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '04, pages 155–165, New York, NY, USA, 2004. ACM.

[12] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional space. In J. Van den Bussche and V. Vianu, editors, *Database Theory — ICDT 2001*, pages 420–434, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[13] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park. Fast algorithms for projected clustering. *SIGMOD Rec.*, 28(2):61–72, June 1999.

[14] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *SIGMOD Rec.*, 27(2):94–105, June 1998.

[15] K. Ahn, G. Cormode, S. Guha, A. McGregor, and A. Wirth. Correlation clustering in data streams. In *International Conference on Machine Learning*, pages 2237–2246, 2015.

[16] A. Amelio and C. Pizzuti. Is normalized mutual information a fair measure for comparing community detection methods? In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1584–1585, Aug 2015.

[17] A. Amini, T. Y. Wah, M. R. Saybani, and S. R. A. S. Yazdi. A study of density-grid based clustering algorithms on data streams. In *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, volume 3, pages 1652–1656, July 2011.

[18] J. Apolloni, G. Leguizamón, and E. Alba. Two hybrid wrapper-filter feature selection algorithms applied to high-dimensional microarray experiments. *Applied Soft Computing*, 38:922 – 932, 2016.

[19] N. Armanfard, J. P. Reilly, and M. Komeili. Local feature selection for data classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(6):1217–1227, June 2016.

[20] A. V. Asimit, E. Furman, and R. Vernic. On a multivariate pareto distribution. *Insurance: Mathematics and Economics*, 46(2):308–316, 2010.

[21] I. Assent, R. Krieger, E. Müller, and T. Seidl. Dusc: Dimensionality unbiased subspace clustering. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 409–414, Oct 2007.

[22] R. Avogadri and G. Valentini. Fuzzy ensemble clustering based on random projections for DNA microarray data analysis. *Artificial Intelligence in Medicine*, 45(2):173 – 183, 2009. Computational Intelligence and Machine Learning in Bioinformatics.

[23] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha. A generalized maximum entropy approach to Bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8(Aug):1919–1986, 2007.

[24] M. Bangert and U. Oelfke. Spherical cluster analysis for beam angle optimization in intensity-modulated radiation therapy treatment planning. *Physics in Medicine & Biology*, 55(19):6023, 2010.

[25] P. M. Barnaghi, M. Bermudez-Edo, R. Tönjes, et al. Challenges for quality of data in smart cities. *J. Data and Information Quality*, 6(2-3):6–1, 2015.

[26] D. Bauer, M. Ray, N. Brändle, and H. Schrom-Feiertag. On extracting commuter information from GPS Motion Data. In *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, page 7, 2008.

[27] Y. Bengio. Deep learning of representations: Looking forward. In *International Conference on Statistical Language and Speech Processing*, pages 1–37. Springer, 2013.

[28] K. Benmouiza and A. Cheknane. Forecasting hourly global solar radiation using hybrid k-means and nonlinear autoregressive neural network models. *Energy Conversion and Management*, 75:561 – 569, 2013.

[29] N. Bhargava, G. Sharma, R. Bhargava, and M. Mathuria. Decision tree analysis on j48 algorithm for data mining. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(6), 2013.

[30] H. Bilen, M. Pedersoli, and T. Tuytelaars. Weakly supervised object detection with convex clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1081–1089, 2015.

[31] B. S. Biswal, P. Mishra, A. Mohapatra, and S. Vipsita. A survey on greedy based algorithms for biclustering of gene expression microarray data. In *2016 International Conference on Information Technology (ICIT)*, pages 124–128, Dec 2016.

[32] R. K. Blashfield and M. S. Aldenderfer. *The Methods and Problems of Cluster Analysis*, pages 447–473. Springer US, Boston, MA, 1988.

[33] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[34] K. Blin, T. Wolf, M. G. Chevrette, X. Lu, C. J. Schwalen, S. A. Kautsar, H. G. Suarez Duran, E. L. De Los Santos, H. U. Kim, M. Nave, et al. AntiSMASH 4.0—improvements in chemistry prediction and gene cluster boundary identification. *Nucleic acids research*, 45(W1):W36–W41, 2017.

[35] I. Borg, P. J. Groenen, and P. Mair. *Applied multidimensional scaling and unfolding*. Springer, 2017.

[36] C. Borgelt. An implementation of the FP-growth algorithm. In *First International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations*, pages 1–5. ACM, 2005.

[37] C. Borgelt. Frequent item set mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6):437–456, 2012.

[38] C. Borgelt and R. Kruse. Induction of association rules: Apriori implementation. In *Proceedings in Computational Statistics*, pages 395–400. Springer, 2002.

[39] T. E. Boult and L. G. Brown. Factorization-based segmentation of motions. In *IEEE Workshop on Visual Motion*, pages 179–186, Oct 1991.

[40] C. Bouveyron and C. Brunet-Saumard. Model-based clustering of high-dimensional data: A review. *Computational Statistics & Data Analysis*, 71:52 – 78, 2014.

[41] K. F. Brill, A. R. Fracasso, and C. M. Bailey. Applying a divisive clustering algorithm to a large ensemble for medium-range forecasting at the weather prediction center. *Weather and Forecasting*, 30(4):873–891, 2015.

[42] M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision (ECCV)*, September 2018.

[43] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16 – 28, 2014. 40th-year commemorative issue.

[44] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan. Deep adaptive image clustering. In *IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[45] S.-H. Chang, P. C. Cosman, and L. B. Milstein. Chernoff-type bounds for the gaussian error function. *IEEE Transactions on Communications*, 59(11):2939–2944, 2011.

[46] M. Charrad and M. Ben Ahmed. Simultaneous clustering: A survey. In S. O. Kuznetsov, D. P. Mandal, M. K. Kundu, and S. K. Pal, editors, *Pattern Recognition and Machine Intelligence*, pages 370–375, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[47] D. Chen, X. Cao, F. Wen, and J. Sun. Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[48] T. Chen, E. Martin, and G. Montague. Robust probabilistic PCA with missing data and contribution analysis for outlier detection. *Computational Statistics & Data Analysis*, 53(10):3706 – 3716, 2009.

[49] W. Chen, Y. Su, Y. Shen, Z. Chen, X. Yan, and W. Y. Wang. How large a vocabulary does text classification need? a variational approach to vocabulary selection. *Proceedings of the NAACL-HLT 2019*, 2019.

[50] X. Chen, J. Z. Huang, Q. Wu, and M. Yang. Subspace weighting co-clustering of gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2017.

[51] C.-H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 84–93, New York, NY, USA, 1999. ACM.

[52] W. Cheng, X. Zhang, F. Pan, and W. Wang. HICC: an entropy splitting-based framework for hierarchical co-clustering. *Knowledge and Information Systems*, 46(2):343–367, 2016.

[53] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra. *Minimum Sum-Squared Residue Co-clustering of Gene Expression Data*, pages 114–125.

[54] M. Cochez and H. Mou. Twister tries: Approximate hierarchical agglomerative clustering for average distance in linear time. In *ACM SIGMOD International Conference on Management of Data*, pages 505–517. ACM, 2015.

[55] J. P. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, Sep 1998.

[56] D. M. Curry. An algorithm for clustering animals by species based upon daily movement. *Procedia Computer Science*, 36:629 – 636, 2014. Complex Adaptive Systems Philadelphia, PA November 3-5, 2014.

[57] I. N. Da Silva, D. H. Spatti, R. A. Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves. Artificial neural networks. *Cham: Springer International Publishing*, 2017.

[58] J. Das, P. Mukherjee, S. Majumder, and P. Gupta. Clustering-based recommender system using principles of voting theory. In *International Conference on Contemporary Computing and Informatics (IC3I)*, pages 230–235, Nov 2014.

[59] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Twentieth Annual Symposium on Computational Geometry*, SCG '04, pages 253–262, New York, NY, USA, 2004. ACM.

[60] R. C. de Amorim and C. Hennig. Recovering the number of clusters in data sets with noise features using feature rescaling factors. *Information Sciences*, 324:126 – 145, 2015.

[61] K. Deb and K. Deb. *Multi-objective Optimization*, pages 403–449. Springer US, Boston, MA, 2014.

[62] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[63] L. Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, Nov 2012.

[64] Z. Deng, K.-S. Choi, Y. Jiang, J. Wang, and S. Wang. A survey on soft subspace clustering. *Information Sciences*, 348:84–106, 2016.

[65] H. Derksen. Hilbert series of subspace arrangements. *Journal of Pure and Applied Algebra*, 209(1):91 – 98, 2007.

[66] H. Derksen, Y. Ma, W. Hong, and J. Wright. Segmentation of multivariate mixed data via lossy coding and compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9):15461562, 2007.

[67] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. In *Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 1117–1126, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics.

[68] N. Dhanachandra, K. Manglem, and Y. J. Chanu. Image segmentation using k -means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 54:764 – 771, 2015.

[69] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017.

[70] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *9th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pages 89–98, 2003.

[71] R. Díaz-Uriarte and S. Alvarez de Andrés. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):3, Jan 2006.

[72] G. Dong and J. Bailey. Overview of contrast data mining as a field and preview of an upcoming book. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 1141–1146, Dec 2011.

[73] D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *National Academy of Sciences*, 100(10):5591–5596, 2003.

[74] H. Edelsbrunner and H. Wagner. Topological data analysis with bregman divergences. *arXiv preprint arXiv:1607.06274*, 2016.

[75] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *National Academy of Sciences*, 95(25):14863–14868, 1998.

[76] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781, 2013.

[77] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[78] P. F. Evangelista, M. J. Embrechts, and B. K. Szymanski. Taming the curse of dimensionality in kernels and novelty detection. In A. Abraham, B. de Baets, M. Köppen, and B. Nickolay, editors, *Applied Soft Computing Technologies: The Challenge of Complexity*, pages 425–438, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[79] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing*, 2(3):267–279, 2014.

[80] F. Fahiman, S. M. Erfani, S. Rajasegarar, M. Palaniswami, and C. Leckie. Improving load forecasting based on deep learning and k-shape clustering. In *2017 International Joint Conference on Neural Networks*, pages 4134–4141, May 2017.

[81] D. Feldman, M. Schmidt, and C. Sohler. Turning big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering. In *Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pages 1434–1453, Philadelphia, PA, USA, 2013. Society for Industrial and Applied Mathematics.

[82] L. Ferrari, A. Rosi, M. Mamei, and F. Zambonelli. Extracting urban patterns from location-based social networks. In *ACM SIGSPATIAL Intl. Workshop on Location-Based Social Networks*, LBSN '11, pages 9–16, 2011.

[83] V. Franzoni, A. Milani, S. Pallottelli, C. H. C. Leung, and Yuanxi Li. Context-based image semantic similarity. In *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 1280–1284, Aug 2015.

[84] M. H. Freitas, F. L. C. Pádua, and G. T. d. Assis. Object-based image retrieval using local feature extraction and relevance feedback. *International Journal of Computer Applications*, 2013.

[85] L. Gao, J. Song, X. Liu, J. Shao, J. Liu, and J. Shao. Learning in high-dimensional multimedia data: the state of the art. *Multimedia Systems*, 23(3):303–313, 2017.

[86] Z. Gao, Y. Fan, K. Niu, and Z. Ying. Mr-mafia: Parallel subspace clustering algorithm based on mapreduce for large multi-dimensional datasets. In *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 257–262, Jan 2018.

[87] M. Gardner and S. Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14):2627 – 2636, 1998.

[88] S. Goil, H. Nagesh, and A. Choudhary. MAFIA: Efficient and scalable subspace clustering for very large data sets. In *Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 443–452. ACM, 1999.

[89] P. Goyal, S. Kumari, S. Singh, V. Kishore, S. S. Balasubramaniam, and N. Goyal. A parallel framework for grid-based bottom-up subspace clustering. In *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 331–340, Oct 2016.

[90] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007.

[91] Q. Gu, Z. Li, and J. Han. Generalized fisher score for feature selection. *CoRR*, abs/1202.3725, 2012.

[92] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[93] F. Gullo, C. Domeniconi, and A. Tagarelli. Projective clustering ensembles. *Data Mining and Knowledge Discovery*, 26(3):452–511, May 2013.

[94] D. E. Gustafson and W. C. Kessel. Fuzzy clustering with a fuzzy covariance matrix. In *1978 IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes*, pages 761–766, Jan 1978.

[95] C. Hachenberg and T. Gottron. Locality sensitive hashing for scalable structural classification and clustering of web documents. In *22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 359–368, New York, NY, USA, 2013. ACM.

[96] M. Hajja, M. Hayajneh, B. Nguyen, and S. Shaqaqha. Distances from the vertices of a regular simplex. *Results in Mathematics*, 72(1-2):633–648, 2017.

[97] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *ACM Sigmod Record*, 29(2):1–12, 2000.

[98] D. Hanisch, A. Zien, R. Zimmer, and T. Lengauer. Co-clustering of biological networks and gene expression data. *Bioinformatics*, 18(suppl_1):S145–S154, 2002.

[99] T. C. Havens and J. C. Bezdek. An efficient formulation of the improved visual assessment of cluster tendency (iVAT) algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):813–822, May 2012.

[100] B. Hayes, J. Gruber, and M. Prodanovic. Short-term load forecasting at the local level using smart meter data. In *2015 IEEE Eindhoven PowerTech*, pages 1–6, June 2015.

[101] M. Hegland. The apriori algorithm–a tutorial. In *Mathematics and computation in imaging science and information processing*, pages 209–262. World Scientific, 2007.

[102] A. Hinneburg and H.-H. Gabriel. Denclue 2.0: Fast clustering based on kernel density estimation. In *Advances in Intelligent Data Analysis VII*, pages 70–80. Springer, 2007.

[103] L. Hollenstein and R. Purves. Exploring place through user-generated content: Using Flickr tags to describe city cores. *Jnl. of Spatial Information Science*, (1):21–48, 2015.

[104] S.-Y. Hsieh and Y.-C. Chou. A faster cDNA microarray gene expression data classifier for diagnosing diseases. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 13(1):43–54, Jan. 2016.

[105] K. Huang, Y. Ma, and R. Vidal. Minimum effective dimension for mixtures of subspaces: a robust GPCA algorithm and its applications. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004.*, volume 2, pages II–631–II–638 Vol.2, June 2004.

[106] Q. Huang, J. Feng, Y. Zhang, Q. Fang, and W. Ng. Query-aware locality-sensitive hashing for approximate nearest neighbor search. *VLDB Endowment*, 9(1):1–12, Sept. 2015.

[107] N. Ichimura. Motion segmentation based on factorization method and discriminant criterion. In *Seventh IEEE International Conference on Computer Vision*, volume 1, pages 600–605 vol.1, 1999.

[108] R. Indhumathi and S. Sathiyabama. Reducing and clustering high dimensional data through principal component analysis. *International Journal of Computer Applications*, 11(8):1–4, 2010.

[109] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[110] J. Irani, N. Pise, and M. Phatak. Clustering techniques and the similarity measures used in clustering: a survey. *International Journal of Computer Applications*, 134(7):9–14, 2016.

[111] T. B. Iredale, S. M. Erfani, and C. Leckie. An efficient visual assessment of cluster tendency tool for large-scale time series data sets. In *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8, July 2017.

[112] A. J. Izenman. *Linear Discriminant Analysis*, pages 237–280. Springer New York, New York, NY, 2008.

[113] G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. In *Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 538–543, New York, NY, USA, 2002. ACM.

[114] D. Jiang, C. Tang, and A. Zhang. Cluster analysis for gene expression data: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386, Nov 2004.

[115] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami. An information framework for creating a smart city through internet of things. *Internet of Things Journal, IEEE*, 1(2):112–121, 2014.

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

[117] I. T. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. *Phil. Trans. R. Soc. A*, 374(2065):20150202, 2016.

[118] R. Jonker and T. Volgenant. Improving the hungarian assignment algorithm. *Operations Research Letters*, 5(4):171–175, 1986.

[119] I. Jurisica. Context-based similarity applied to retrieval of relevant cases. Technical report, AAAI Fall Symposium Series on Relevance, 1994.

[120] K. Kailing, H.-P. Kriegel, and P. Kröger. Density-connected subspace clustering for high-dimensional data. In *SIAM International Conference on Data Mining*, pages 246–256.

[121] A. Kaltenbrunner, R. Meza, J. Grivolla, J. Codina, and R. Banchs. Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system. *Pervasive and Mobile Computing*, 6(4):455–466, 2010.

[122] K. Kanatani. Motion segmentation by subspace separation and model selection. In *Eighth IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 586–591 vol.2, 2001.

[123] J. A. Kelly and J. P. Clinch. Influence of varied parking tariffs on parking occupancy levels by trip purpose. *Transport Policy*, 13(6):487–495, 2006.

[124] G. Kerr, H. Ruskin, M. Crane, and P. Doolan. Techniques for clustering gene expression data. *Computers in Biology and Medicine*, 38(3):283 – 293, 2008.

[125] S. Kisilevich, F. Mansmann, and D. Keim. P-dbscan: A density based clustering algorithm for exploration and analysis of attractive areas using collections of geotagged photos. In *First International Conference and Exhibition on Computing for Geospatial Research Application*, COM.Geo '10, pages 38:1–38:4, New York, NY, USA, 2010. ACM.

[126] J. Kléma, F. Malinka, and F. Zelezny. Semantic biclustering: a new way to analyze and interpret gene expression data. *Bioinformatics Research and Applications, Minsk, Belarus, Springer*, pages 332–3, 2016.

[127] H. Koga, T. Ishibashi, and T. Watanabe. Fast agglomerative hierarchical clustering algorithm using locality-sensitive hashing. *Knowledge and Information Systems*, 12(1):25–53, May 2007.

[128] H.-P. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *Knowledge Discovery from Data*, 3(1):1–58, 2009.

[129] K. M. Kumar and A. R. M. Reddy. A fast dbscan clustering algorithm by accelerating neighbor searching using groups method. *Pattern Recognition*, 58:39 – 48, 2016.

[130] R. R. Larson. Introduction to information retrieval. *Journal of the American Society for Information Science and Technology*, 61(4):852–853, 2010.

[131] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *26th Annual International Conference on Machine Learning*, ICML '09, pages 609–616, New York, NY, USA, 2009. ACM.

[132] T.-W. Lee. *Independent Component Analysis*, pages 27–66. Springer US, Boston, MA, 1998.

[133] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764 – 766, 2013.

[134] B. Li and L. Han. Distance weighted cosine similarity measure for text classification. In H. Yin, K. Tang, Y. Gao, F. Klawonn, M. Lee, T. Weise, B. Li, and X. Yao, editors, *Intelligent Data Engineering and Automated Learning – IDEAL 2013*, pages 611–618, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[135] S. Li, K. Li, and Y. Fu. Temporal subspace clustering for human motion segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[136] Y. Li and J. Zhu. L1-norm quantile regression. *Journal of Computational and Graphical Statistics*, 2012.

[137] A. Liaw and M. Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[138] B. Liu, Y. Xia, and P. S. Yu. Clustering through decision tree construction. In *Ninth International Conference on Information and Knowledge Management*, CIKM '00, pages 20–29, New York, NY, USA, 2000. ACM.

[139] G. Liu and S. Yan. Latent low-rank representation for subspace segmentation and feature extraction. In *International Conference on Computer Vision (ICCV)*, pages 1615–1622, 2011.

[140] H. Liu, T. Liu, J. Wu, D. Tao, and Y. Fu. Spectral ensemble clustering. In *21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 715–724, New York, NY, USA, 2015. ACM.

[141] Y. Liu, Q. Gu, J. P. Hou, J. Han, and J. Ma. A network-assisted co-clustering algorithm to discover cancer subtypes based on gene expression. *BMC Bioinformatics*, 15(1):37, Feb 2014.

[142] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Intelligent probing for locality sensitive hashing: Multi-probe lsh and beyond. *VLDB Endowment*, 10(12):2021–2024, Aug. 2017.

[143] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[144] P. Mahanta, H. A. Ahmed, J. K. Kalita, and D. K. Bhattacharyya. Discretization in gene expression data analysis: A selected survey. In *Second International Conference on Computational Science, Engineering and Information Technology*, CCSEIT '12, pages 69–75, New York, NY, USA, 2012. ACM.

[145] M. Mamei, A. Rosi, and F. Zambonelli. Automatic analysis of geotagged photos for intelligent tourist services. In *2010 Sixth International Conference on Intelligent Environments*, pages 146–151, July 2010.

[146] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[147] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob. Big iot data analytics: Architecture, opportunities, and open research challenges. *IEEE Access*, 5:5247–5261, 2017.

[148] K. G. Mehrotra, C. K. Mohan, and H. Huang. *Clustering-Based Anomaly Detection Approaches*, pages 41–55. Springer International Publishing, Cham, 2017.

[149] Melbourne. Parking bay arrivals and departures 2016. https://data.melbourne.vic.gov.au/Transport-Movement/On-street-Car-Parking-Sensor-Data-2016/dj7e-rdx9, 2016.

[150] Melbourne. Parking bay arrivals and departures 2014. https://data.melbourne.vic.gov.au/Transport-Movement/Parking-Events-2014/mq3i-cbxd, 2018.

[151] A. Melo and H. Paulheim. Local and global feature selection for multilabel classification with binary relevance. *Artificial Intelligence Review*, May 2017.

[152] F. Meng, Y. Gao, and R. Wang. A novel automatic context-based similarity metric for local outlier detection tasks. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 983–990, Nov 2018.

[153] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin. Advances in pre-training distributed word representations. *CoRR*, abs/1712.09405, 2017.

[154] A. Millonig and G. Gartner. Shadowing - tracking - interviewing: How to explore human spatio-temporal behaviour patterns. In *2nd Workshop on Behaviour Monitoring and Interpretation, BMI'08, Kaiserslautern, Germany, September 23, 2008*, pages 1–14, 2008.

[155] M. Moshtaghi, T. C. Havens, J. C. Bezdek, L. Park, C. Leckie, S. Rajasegarar, J. M. Keller, and M. Palaniswami. Clustering ellipses for anomaly detection. *Pattern Recognition*, 44(1):55 – 69, 2011.

[156] M. Moshtaghi, S. Rajasegarar, C. Leckie, and S. Karunasekera. An efficient hyperellipsoidal clustering algorithm for resource-constrained environments. *Pattern Recognition*, 44(9):2197 – 2209, 2011. Computer Analysis of Images and Patterns.

[157] H.-J. Mucha. On validation of hierarchical clustering. In R. Decker and H. J. Lenz, editors, *Advances in Data Analysis*, pages 115–122, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[158] E. Müller, I. Assent, S. Günnemann, P. Gerwert, M. Hannen, T. Jansen, and T. Seidl. A framework for evaluation and exploration of clustering algorithms in subspaces of high dimensional databases. *Datenbanksysteme für Business, Technologie und Web (BTW)*, 2011.

[159] E. Müller, I. Assent, S. Günnemann, and T. Seidl. Opensubspace: An open source framework for evaluation and exploration of subspace clustering algorithms in weka. In *Proceedings of OSDM 2009*, pages 1–12, 2009. Kun udgivet elektronisk.

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

[161] R. A. Musheer, C. K. Verma, and N. Srivastava. Novel machine learning approach for classification of high-dimensional microarray data. *Soft Computing*, 23(24):13409–13421, Dec 2019.

[162] L. Nebeský. The minimum degree and connectivity of a graph. In Y. Alavi and D. R. Lick, editors, *Theory and Applications of Graphs*, pages 412–419, Berlin, Heidelberg, 1978. Springer Berlin Heidelberg.

[163] N. Nesa, T. Ghosh, and I. Banerjee. Outlier detection in sensed data using statistical learning models for iot. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, April 2018.

[164] D. J. Ozer. Correlation and the coefficient of determination. *Psychological Bulletin*, 97(2):307, 1985.

[165] J. Paparrizos and L. Gravano. k-shape: Efficient and accurate clustering of time series. In *ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1855–1870, New York, NY, USA, 2015. ACM.

[166] L. A. F. Park, J. C. Bezdek, C. Leckie, R. Kotagiri, J. Bailey, and M. Palaniswami. Visual assessment of clustering tendency for incomplete data. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3409–3422, Dec 2016.

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

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

[169] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[170] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018.

[171] S. Petrovic. A comparison between the silhouette index and the Davies-Bouldin index in labelling IDS clusters. In *Eleventh Nordic Workshop of Secure IT Systems*, pages 53–64. sn, 2006.

[172] D. Pfitzner, R. Leibbrandt, and D. Powers. Characterization and evaluation of similarity measures for pairs of clusterings. *Knowledge and Information Systems*, 19(3):361, Jul 2008.

[173] R. Pless and R. Souvenir. A survey of manifold learning for images. *IPSJ Transactions on Computer Vision and Applications*, 1:83–94, 2009.

[174] B. Pontes, R. Giráldez, and J. S. Aguilar-Ruiz. Biclustering on expression data: A review. *Journal of Biomedical Informatics*, 57:163 – 180, 2015.

[175] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2011.

[176] A. Prelić, S. Bleuler, P. Zimmermann, A. Wille, P. Bühlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler. A systematic comparison and evaluation of biclustering methods for gene expression data. *Bioinformatics*, 22(9):1122–1129, 2006.

[177] D. Puschmann, P. Barnaghi, and R. Tafazolli. Adaptive clustering for dynamic iot data streams. *IEEE Internet of Things Journal*, 4(1):64–74, Feb 2017.

[178] G. Qian, S. Sural, Y. Gu, and S. Pramanik. Similarity between euclidean and cosine angle distance for nearest neighbor queries. In *2004 ACM Symposium on Applied Computing*, SAC '04, pages 1232–1237, New York, NY, USA, 2004. ACM.

[179] S. Rajasegarar, J. C. Bezdek, M. Moshtaghi, C. Leckie, T. C. Havens, and M. Palaniswami. Measures for clustering and anomaly detection in sets of higher dimensional ellipsoids. In *2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, June 2012.

[180] S. Rajasegarar, A. Gluhak, M. A. Imran, M. Nati, M. Moshtaghi, C. Leckie, and M. Palaniswami. Ellipsoidal neighbourhood outlier factor for distributed anomaly detection in resource constrained networks. *Pattern Recognition*, 47(9):2867 – 2879, 2014.

[181] D. S. Rajput. MAFCA-S: A subspace clustering technique for high dimensional dataset. *International Journal of Advanced Information in Science and Technology*, 2014.

[182] D. S. Rajput, P. K. Singh, and M. Bhattacharya. *PROFIT: A Projected Clustering Technique*, pages 51–70. Springer International Publishing, Cham, 2015.

[183] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[184] R. V. Rao. *A Novel Weighted Euclidean Distance-Based Approach*, pages 159–191. Springer, London, 2013.

[185] P. Rebeschini, R. Van Handel, et al. Can local particle filters beat the curse of dimensionality? *The Annals of Applied Probability*, 25(5):2809–2866, 2015.

[186] S. Romano, J. Bailey, V. Nguyen, and K. Verspoor. Standardized mutual information for clustering comparisons: one step further in adjustment for chance. In *International Conference on Machine Learning*, pages 1143–1151, 2014.

[187] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[188] S. Sagiroglu and D. Sinanc. Big data: A review. In *2013 International Conference on Collaboration Technologies and Systems (CTS)*, pages 42–47, May 2013.

[189] T. Sajana, C. S. Rani, and K. Narayana. A survey on clustering techniques for big data mining. *Indian journal of Science and Technology*, 9(3):1–12, 2016.

[190] M. Salehi, C. A. Leckie, M. Moshtaghi, and T. Vaithianathan. A relevance weighted ensemble model for anomaly detection in switching data streams. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 461–473. Springer, 2014.

[191] V. Satuluri and S. Parthasarathy. Bayesian locality sensitive hashing for fast similarity search. *VLDB Endowment*, 5(5):430–441, Jan. 2012.

[192] D. Schnitzer, A. Flexer, M. Schedl, and G. Widmer. Local and global scaling reduce hubs in space. *Journal of Machine Learning Research*, 13(Oct):2871–2902, 2012.

[193] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu. Dbscan revisited, revisited: Why and how you should (still) use dbscan. *ACM Trans. Database Syst.*, 42(3):19:1–19:21, July 2017.

[194] G. A. Seber and A. J. Lee. *Linear regression analysis*, volume 329. John Wiley & Sons, 2012.

[195] S. Selva Kumar and H. Hannah Inbarani. Analysis of mixed c-means clustering approach for brain tumour gene expression data. *International Journal of Data Analysis Techniques and Strategies*, 5(2):214–228, 2013. PMID: 53682.

[196] A. Shilton, S. Rajasegarar, C. Leckie, and M. Palaniswami. DP1SVM: A dynamic planar one-class support vector machine for internet of things environment. In *International Conference on Recent Advances in Internet of Things (RIoT)*, pages 1–6. IEEE, 2015.

[197] P. Shrivastava and H. Gupta. A review of density-based clustering in spatial data. *International Journal of Advanced Computer Research*, 2(3):200, 2012.

[198] A. Siahkamari, V. Saligrama, D. A. Castañón, and B. Kulis. Learning bregman divergences. *ArXiv*, abs/1905.11545, 2019.

[199] M. P. Sinka and D. W. Corne. A large benchmark dataset for web document clustering. *Soft computing systems: design, management and applications*, 87:881–890, 2002.

[200] D. B. Skillicorn. *Understanding high-dimensional spaces*. Springer Science & Business Media, 2012.

[201] P. Slavik. A tight analysis of the greedy algorithm for set cover. *Journal of Algorithms*, 25(2):237 – 254, 1997.

[202] M. Soltanolkotabi, E. J. Candes, et al. A geometric analysis of subspace clustering with outliers. *The Annals of Statistics*, 40(4):2195–2238, 2012.

[203] M. Soltanolkotabi, E. Elhamifar, and E. J. Candès. Robust subspace clustering. *The Annals of Statistics*, 42(2):669–699, 2014.

[204] D. Steinley and M. J. Brusco. A note on the expected value of the Rand index. *British Journal of Mathematical and Statistical Psychology*, 71(2):287–299, 2018.

[205] S. M. Stigler. The asymptotic distribution of the trimmed mean. *The Annals of Statistics*, 1(3):472–477, 1973.

[206] V. C. Storey and I.-Y. Song. Big data technologies and management: What conceptual modeling can do. *Data & Knowledge Engineering*, 108:50 – 67, 2017.

[207] G. Strang. *Introduction to linear algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA, 1993.

[208] C. Swamy. Correlation clustering: Maximizing agreements via semidefinite programming. In *Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 526–527, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[209] N. Tajunisha and V. Saravanan. An increased performance of clustering high dimensional data using principal component analysis. In *2010 First International Conference on Integrated Intelligent Computing*, pages 17–21, Aug 2010.

[210] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to data mining*. Pearson Education India, 2016.

[211] R. Tang and S. Fong. Clustering big iot data by metaheuristic optimized mini-batch and parallel partition-based dgc in hadoop. *Future Generation Computer Systems*, 86:1395 – 1412, 2018.

[212] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[213] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Comput.*, 11(2):443–482, Feb. 1999.

[214] W.-C. Tjhi and L. Chen. Flexible fuzzy co-clustering with feature-cluster weighting. In *Ninth International Conference on Control, Automation, Robotics and Vision, 2006*, pages 1–6, 2006.

[215] S. Trivedi, Z. A. Pardos, and N. T. Heffernan. The utility of clustering in prediction tasks. *arXiv preprint arXiv:1509.06163*, 2015.

[216] C.-F. Tsai. Combining cluster analysis with classifier ensembles to predict financial distress. *Information Fusion*, 16:46 – 58, 2014. Special Issue on Information Fusion in Hybrid Intelligent Fusion Systems.

[217] M. C. Tsakiris and R. Vidal. Algebraic clustering of affine subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2):482–489, Feb 2018.

[218] D. Tzikas, A. Likas, and N. Galatsanos. Local feature selection for the relevance vector machine using adaptive kernel learning. In C. Alippi, M. Polycarpou, C. Panayiotou, and G. Ellinas, editors, *Artificial Neural Networks – ICANN 2009*, pages 50–59, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[219] A. Vahdat, M. Heywood, and N. Zincir-Heywood. Bottom-up evolutionary subspace clustering. In *2010 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2010.

[220] L. Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[221] S. van der Spek. Spatial metro–tracking pedestrians in historic city centres. *Research in Urbanism Series*, 1(1):77–97, 2008.

[222] R. Vidal. Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68, 2011.

[223] R. Vidal, Y. Ma, and S. Sastry. Generalized principal component analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1945–1959, Dec 2005.

[224] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: Is a correction for chance necessary? In *26th Annual International Conference on Machine Learning*, ICML '09, pages 1073–1080, New York, NY, USA, 2009. ACM.

[225] A. L. Vlek, B. S. Cooper, T. Kypraios, A. Cox, J. D. Edgeworth, and O. T. Auguet. Clustering of antimicrobial resistance outbreaks across bacterial species in the intensive care unit. *Clinical infectious diseases*, 57(1):65–76, 2013.

[226] Q. Wang and G. Chen. Fuzzy soft subspace clustering method for gene co-expression network analysis. *International Journal of Machine Learning and Cybernetics*, 8(4):1157–1165, Aug 2017.

[227] S. Wang, B. Tu, C. Xu, and Z. Zhang. Exact subspace clustering in linear time. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[228] S. Wang, X. Yuan, T. Yao, S. Yan, and J. Shen. Efficient subspace segmentation via quadratic programming. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, pages 519–524. AAAI Press, 2011.

[229] Y. Wang, L. Kung, and T. A. Byrd. Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations. *Technological Forecasting and Social Change*, 126:3 – 13, 2018.

[230] Y.-X. Wang and H. Xu. Noisy sparse subspace clustering. *The Journal of Machine Learning Research*, 17(1):320–360, 2016.

[231] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, volume 98, pages 194–205, 1998.

[232] K.-G. Woo, J.-H. Lee, M.-H. Kim, and Y.-J. Lee. Findit: a fast and intelligent subspace clustering algorithm using dimension voting. *Information and Software Technology*, 46(4):255 – 271, 2004.

[233] Y. Wu, Z. Zhang, T. S. Huang, and J. Y. Lin. Multibody grouping via orthogonal subspace decomposition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 2, pages II–252–II–257 vol.2, 2001.

[234] S. Xia, Z. Xiong, Y. Luo, WeiXu, and G. Zhang. Effectiveness of the Euclidean distance in high dimensional spaces. *Optik - International Journal for Light and Electron Optics*, 126(24):5614 – 5619, 2015.

[235] M. Xu, P. Watanachaturaporn, P. K. Varshney, and M. K. Arora. Decision tree regression for soft classification of remote sensing data. *Remote Sensing of Environment*, 97(3):322–336, 2005.

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

[237] Z. Xu and M. Xia. Distance and similarity measures for hesitant fuzzy sets. *Information Sciences*, 181(11):2128 – 2138, 2011.

[238] X. Yan, X. J. Zhou, and J. Han. Mining closed relational graphs with connectivity constraints. In *Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 324–333, New York, NY, USA, 2005. ACM.

[239] A. Y. Yang, S. R. Rao, and Y. Ma. Robust statistical estimation and segmentation of multiple subspaces. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, pages 99–99, June 2006.

[240] Q. Yang, H. Wu, C.-Y. Guo, and C. S. Fox. Analyze multivariate phenotypes in genetic association studies by combining univariate association tests. *Genetic Epidemiology*, 34(5):444–454, 2010.

[241] Y. Ye, Q. Wu, J. Z. Huang, M. K. Ng, and X. Li. Stratified sampling for feature subspace selection in random forests for high dimensional data. *Pattern Recognition*, 46(3):769 – 787, 2013.

[242] C. You, D. Robinson, and R. Vidal. Scalable sparse subspace clustering by orthogonal matching pursuit. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[243] X. Yu, G. Yu, and J. Wang. Clustering cancer gene expression data by projective clustering ensemble. *PLOS ONE*, 12(2):1–21, 02 2017.

[244] T. Zhang, A. Szlam, and G. Lerman. Median k-flats for hybrid linear modeling with many outliers. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 234–241, Sept 2009.

[245] Z. Zhang, J. Wang, and H. Zha. Adaptive manifold learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(2):253–265, Feb 2012.

[246] J. Zhao and Q. Jiang. Probabilistic PCA for t distributions. *Neurocomputing*, 69(16):2217 – 2226, 2006. Brain Inspired Cognitive Systems.

[247] Q. Zhao et al. Knee point detection in BIC for detecting the number of clusters. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 664–673. Springer, 2008.

[248] B. Zhu, A. Mara, and A. Mozo. Clus: Parallel subspace clustering algorithm on spark. In T. Morzy, P. Valduriez, and L. Bellatreche, editors, *New Trends in Databases and Information Systems*, pages 175–185, Cham, 2015. Springer International Publishing.

[249] P. Zhu, W. Zhu, Q. Hu, C. Zhang, and W. Zuo. Subspace clustering guided unsupervised feature selection. *Pattern Recognition*, 66:364 – 374, 2017.

Appendix A

Supplementary Material

In this section, for completeness and for the purpose of reproducibility, we present the supplementary material, including the results and ground truth that are used in Chapters 3-6.

date	hour	label	predict
2014-06-06	22	1	1
2014-06-06	23	1	1
2014-06-06	midnight	1	0
2014-06-13	22	1	1
2014-06-13	23	1	1
2014-06-13	midnight	1	0
2014-06-20	22	1	1
2014-06-20	23	1	0
2014-06-20	midnight	1	0
2014-06-27	22	1	1
2014-06-27	23	1	0
2014-06-27	midnight	1	0
2014-07-04	22	1	1
2014-07-04	23	1	0
2014-07-04	midnight	1	0
2014-07-11	22	1	1
2014-07-11	23	1	0
2014-07-11	midnight	1	0
2014-07-18	22	1	0
2014-07-18	23	1	0
2014-07-18	midnight	1	0

2014-07-25	22	1	0
2014-07-25	23	1	0
2014-07-25	midnight	1	0
2014-08-01	22	1	1
2014-08-01	23	1	0
2014-08-01	midnight	1	0
2014-08-08	22	1	1
2014-08-08	23	1	1
2014-08-08	midnight	1	0
2014-08-15	22	1	1
2014-08-15	23	1	1
2014-08-15	midnight	1	0
2014-08-22	22	1	1
2014-08-22	23	1	1
2014-08-22	midnight	1	0
2014-08-29	22	1	1
2014-08-29	23	1	1
2014-08-29	midnight	1	0
2014-09-05	22	1	1
2014-09-05	23	1	1
2014-09-05	midnight	1	1
2014-09-12	22	1	1
2014-09-12	23	1	1
2014-09-12	midnight	1	1
2014-09-19	22	1	1
2014-09-19	23	1	1
2014-09-19	midnight	1	1
2014-09-26	22	1	1
2014-09-26	23	1	1
2014-09-26	midnight	1	1
2014-10-03	22	1	1
2014-10-03	23	1	1
2014-10-03	midnight	1	1
2014-10-10	22	1	1
2014-10-10	23	1	1
2014-10-10	midnight	1	1

2014-10-17	22	1	1
2014-10-17	23	1	1
2014-10-17	midnight	1	0
2014-10-24	22	1	1
2014-10-24	23	1	1
2014-10-24	midnight	1	1
2014-10-31	22	1	1
2014-10-31	23	1	1
2014-10-31	midnight	1	1
2014-11-07	22	1	1
2014-11-07	23	1	1
2014-11-07	midnight	1	0
2014-11-14	22	1	1
2014-11-14	23	1	1
2014-11-14	midnight	1	1
2014-11-21	22	1	1
2014-11-21	23	1	1
2014-11-21	midnight	1	1
2014-11-28	22	1	1
2014-11-28	23	1	1
2014-11-28	midnight	1	1
2014-12-05	22	1	1
2014-12-05	23	1	1
2014-12-05	midnight	1	1
2014-12-12	22	1	1
2014-12-12	23	1	1
2014-12-12	midnight	1	0
2014-12-19	22	1	1
2014-12-19	23	1	1
2014-12-19	midnight	1	0
2014-12-26	22	1	1
2014-12-26	23	1	1
2014-12-26	midnight	1	1

Table A.1 Ground truth and predicted anomalies during evening time of Fridays from 1st June to 31st December 2014. The table was used in Section 3.4.4.

Table A.2 Deployment map of sensors with categories of the locations. Locations extracted from the website of Pedestrian Counting System of the City of Melbourne [1]. The locations and their classifications were referenced in Chapter 3 and Chapter 6.

ID	Location	Location Type				
		Train Station	Shopping Center	Restaurant/Cafe	Tourist Attraction	Office
1	The Arts Centre				X	
2	St Kilda Rd-Alexandra Gardens	X			X	
3	Birrarung Marr				X	
4	Princes Bridge				X	
5	Flinders Street Station Underpass	X		X		
6	Flinders St-Swanston St (West)	X				
7	Flinders St-Elizabeth St (East)	X				
8a	Flinders St-Spring St (West)					X
8b	Flinders St-Spark La					
9a	Collins Place (South)			X		X
9b	Collins Place (North)			X		X
10	City Square			X	X	
11a	Bourke Street Mall (South)		X	X	X	
11b	Bourke Street Mall (North)		X	X	X	
12	Chinatown-Swanston St (North)		X	X		
13	Chinatown-Lt Bourke St (South)		X	X		
14a	Melbourne Central	X	X	X		
14b	State Library		X		X	
15	Flagstaff Station	X				
16a	Spencer St-Collins St (South)			X		X
16b	Spencer St-Collins St (North)			X		X
17	Southern Cross Station	X		X		
18	New Quay			X	X	
19	Waterfront City				X	
20	Melbourne Convention Exhibition Centre					X
21	Bourke St-Russell St (West)			X		X
22	Webb Bridge					X
23	Victoria Point				X	
24	Sandridge Bridge				X	
25	QV Market-Elizabeth St (West)			X		
26	QV Market-Peel St		X			
27	Lonsdale St (South)		X	X		
28	Lonsdale St-Spring St (West)	X			X	X
29a	Lygon St (East)			X		
29b	Lygon St (West)			X		
30	Southbank			X	X	
31	Queen St (West)					X
32	Alfred Place					
32	Town Hall(West)			X	X	
33	Collins St (North)		X			X
34	Grattan St-Swanston St (West)					X
35	Monash Rd-Swanston St (West)					X
36	Tin Alley-Swanston St (West)					X

SWCC	ITCC	EWKM	SCBS	BBAC	label
2	2	1	1	2	0
2	1	2	1	1	0
1	1	1	1	2	0
2	1	2	3	1	0
1	2	1	3	1	0
1	1	2	3	1	0
2	1	1	2	1	0
1	2	2	1	1	0
1	1	1	2	1	0
2	2	1	1	1	0
2	2	2	1	2	0
2	2	1	2	2	0
2	1	1	3	2	0
1	1	2	3	1	0
1	2	2	2	2	0
2	2	1	2	1	0
2	1	2	2	2	0
1	2	2	3	2	0
2	2	2	3	2	0
2	2	1	1	1	0
1	1	2	1	2	0
2	2	1	2	2	0
2	2	2	2	2	0
2	2	1	3	2	0
1	2	1	1	1	0
1	2	2	3	1	0
2	1	1	1	2	0
1	1	1	2	1	0
2	2	1	3	2	0
1	2	1	2	2	0
2	1	2	2	1	0
2	2	1	1	2	0
2	2	1	3	2	0
2	1	2	1	2	0
1	1	1	1	1	0
2	1	1	3	1	0
2	1	2	1	1	0
1	1	1	3	2	0
2	2	1	1	2	0
2	1	1	3	2	0
1	2	2	3	2	0
1	1	1	2	1	0
2	2	1	3	2	0
1	2	2	1	2	0
2	2	1	3	1	0
1	2	2	1	1	0
2	1	2	1	1	0
1	1	2	2	2	0
2	2	1	2	1	0
2	1	2	1	2	0
2	2	1	3	1	0
1	2	1	1	1	0
1	1	1	1	1	0
2	1	2	3	1	0

2	1	2	2	2	0
2	1	2	2	1	0
2	1	1	2	1	0
2	1	2	1	2	0
2	2	1	1	1	0
2	1	1	2	1	0
2	1	2	3	1	0
2	2	1	1	2	0
1	1	2	2	2	0
1	2	2	2	1	0
2	1	1	1	1	1
1	2	1	2	1	1
1	1	2	2	1	1
2	1	2	3	2	1
1	1	1	3	1	1
1	2	2	2	2	1
1	2	1	2	1	1
1	1	1	2	2	1
1	1	2	3	2	1
2	1	2	2	2	1
2	2	1	2	1	1
2	1	1	1	1	1

Table A.3 Clustering results and ground truth of ADE dataset that were discussed in Section 6.2.1

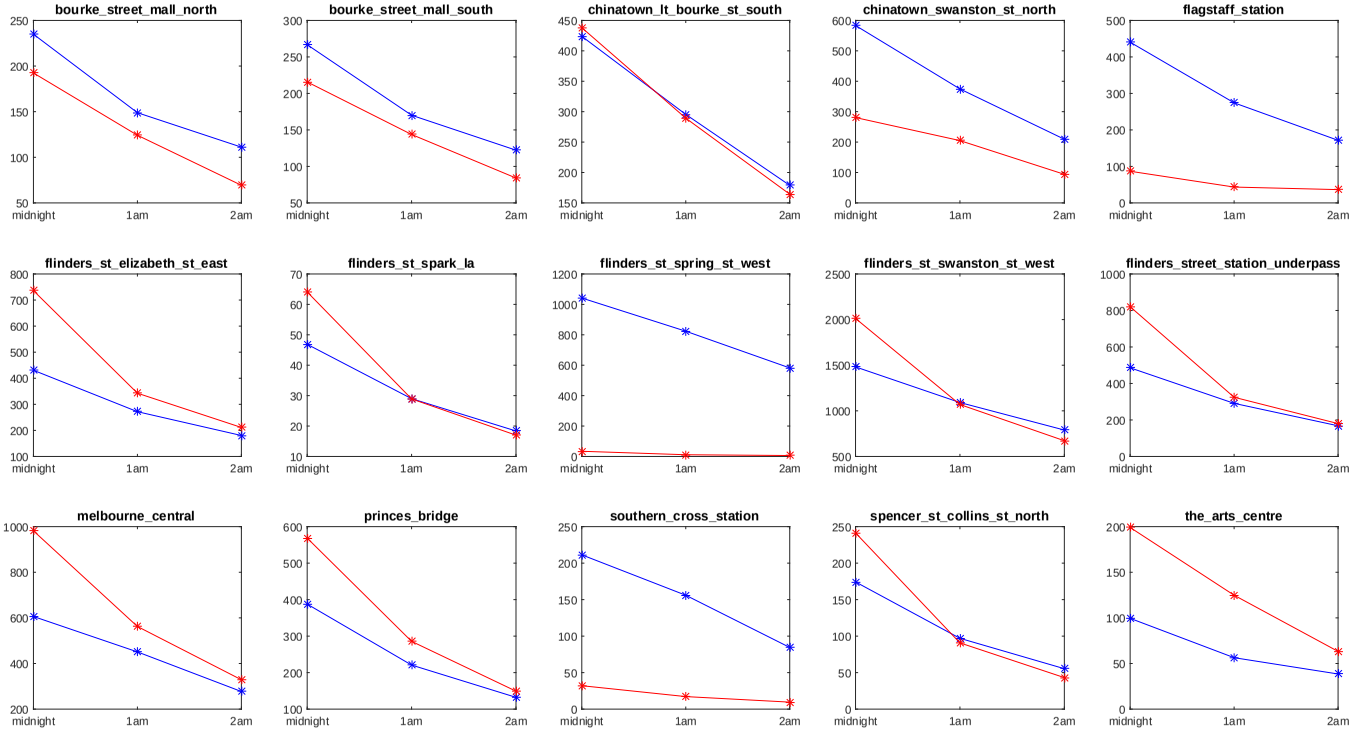
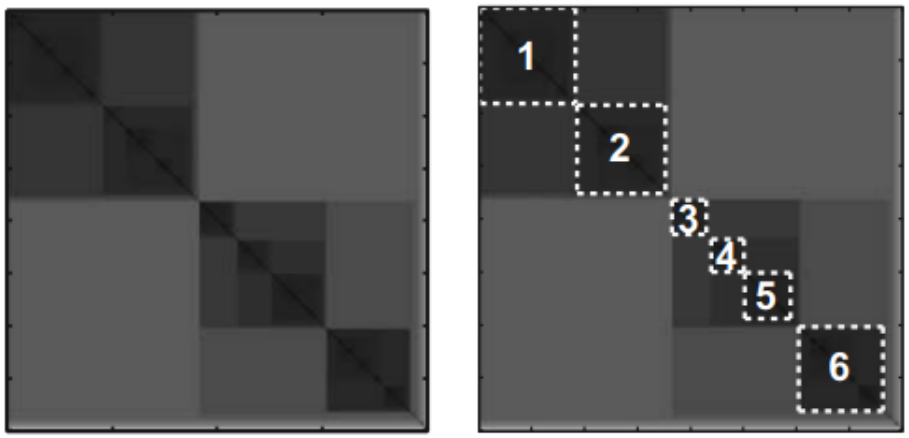
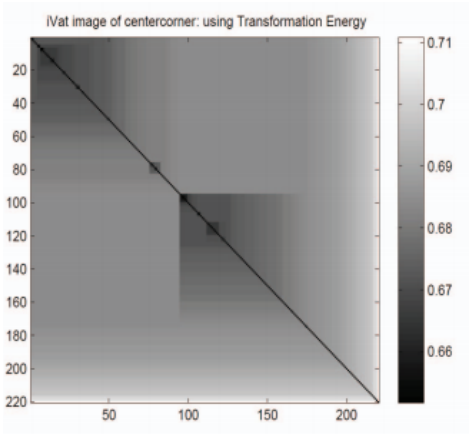


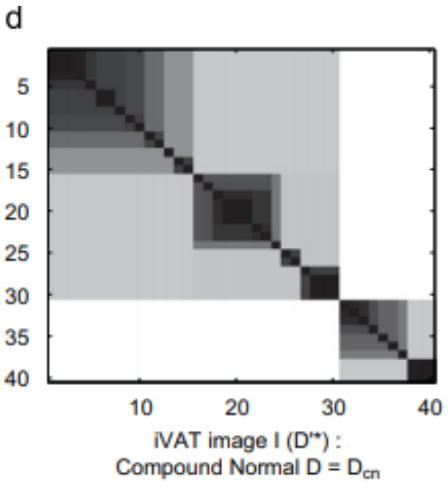
Fig. A.1 Average pedestrian counts during Saturday and Sunday midnights (midnight – 2am) at each location before and after the introduction of the Night Network, as discussed in Section 6.1.2. Blue line and red line correspond to pedestrian counts at 2016 and 2015 respectively.



(a)



(b)



(c)

Fig. A.2 Interpretation of iVAT images from the authors of iVAT, as discussed in Section 6.1.1. (a) An example of an iVAT image in which fairly distinct blocks are discovered. Note that the outer blocks with lower gray intensity are still not considered in [99] (Figure 2(bc)), (b) An example of iVAT image that the authors conclude "fails to reveal any clustering structure" [179] (last subfigure of Figure 6), (c) An example of iVAT image that is considered fragmented by the authors [155] (Figure 9d)

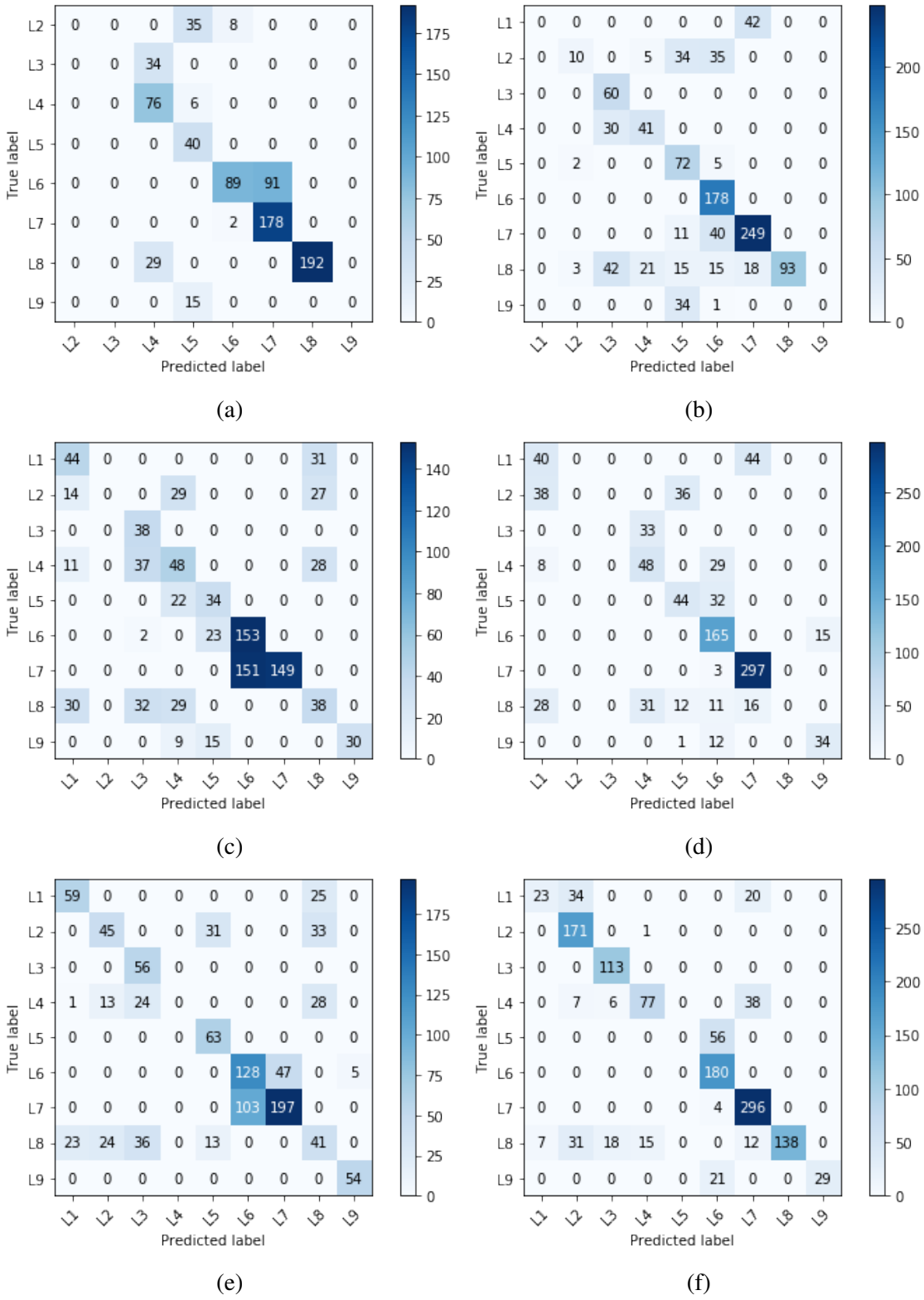


Fig. A.3 Confusion matrices of predicting characteristics of pedestrian activities for (a) Jun 2016 - Jul 2016, (b) Aug 2016 - Sep 2016, (c) Oct 2016 - Nov 2016, (d) Dec 2016 - Jan 2017, (e) Feb 2017 - Mar 2017. This figure provides the full quantitative evaluation results of all periods being analyzed in Section 6.1.1.