Subspace Clustering and Active Learning with Constraints

Hankui Peng, B.S., M.S., M.Res

Submitted for the degree of Doctor of Philosophy

at Lancaster University

September 2020

献给我的父母，感谢你们一直以来在我追求知识的道路上无条件的付出。

To my parents, for their unwavering support in my pursuit of knowledge.

Abstract

Data representations can often be high-dimensional, whether it is due to the large number of collected / recorded features or due to how the data sources (e.g. images, texts) are processed. It is often the case that the main structure of the data can be summarised well in a lower dimensional subspace or multiple lower dimensional subspaces. Subspace clustering addresses the problem of simultaneously uncovering multiple subspace structures in the data and grouping the data according to their underlying subspace structures.

The first contribution of this thesis is the development of a Subspace Clustering with Active Learning (SCAL) framework that is designed for K-Subspace Clustering. This framework allows clustering performance to improve in an effective and efficient manner over time, with the need to query only a small amount of labelling information. It also has the potential to be applied to more general subspace clustering methods, which has been further explored and developed in our next methodological contribution.

The second contribution of this thesis is a unified active learning and constrained clustering framework for spectral-based subspace clustering methods. In this work, we propose a spectral-based subspace clustering methodology named Weighted Sparse Simplex Representation (WSSR). It has been demonstrated to have favourable performance against state-of-the-art spectral-based subspace clustering methods on both synthetic and real data. We also propose a flexible weighting scheme that can incorporate external information into the problem formulation, which leads to a constrained clustering extension of WSSR. We show that it can be applied in conjunction with our previously proposed SCAL strategy when labelling information can be queried sequentially.

The third contribution of this thesis is the development of an algebraic subspace clustering methodology – Minimum Angle Clustering (MAC). It is motivated by the application of clustering Amazon products based on their titles when represented using the TF-IDF matrix, which is both sparse and high-dimensional. The proposed methodology is composed of two stages. In the first stage, it identifies a large number of subspaces in the data through the Reduced Row Echelon Form technique. In the second stage, we propose a new subspace proximity measure to construct an affinity matrix for the formed subspaces before spectral clustering is applied to obtain the final cluster labels. The proposed methodology has been shown to enjoy competitive performance against a number of well-established subspace clustering and document clustering techniques on the application of clustering Amazon product names.

Acknowledgements

This work would not have been made possible without the help and support of many great people around me. First and foremost, I am mostly indebted to and tremendously grateful for my main advisor Dr. Nicos Pavlidis, who played an absolutely essential and hugely instrumental role in my PhD. Nicos, it was my great pleasure to work with you, and have you as my mentor and mainstay throughout the whole journey. Thank you for your thorough guidance with insightful questions and suggestions during our meetings, which always led me to a deeper understanding of my work. Secondly, I would like to thank my advisor Professor Idris Eckley. Thank you for always reminding me of the importance of the big picture, for keeping me on track at times of need, and for providing your words of wisdom on academic matters and otherwise.

I have been incredibly lucky and privileged to have had the opportunity to collaborate with the Data Science Campus (DSC) within the Office for National Statistics (ONS). It was a truly wonderful and unique experience that I relish very much. Special thanks to Ian and Thanasis for introducing me to your interesting projects and for providing me with helpful feedback throughout the whole process.

I would also like to thank all the directors and administrative team of STOR-i, for creating and fostering such a fantastically vibrant and supportive community. Thank you for providing me with the financial support and the opportunity to grow, I could not imagine a better place to do my PhD. In particular, I cannot overstate my gratitude for Professor Jonathan Tawn, who is not my official ‘advisor’, but fully fulfils every sense of the word. Jon, thank you for your invaluable time and patience during all the one-to-one tutoring sessions in the MRes year; thank you for being so extremely attentive, supportive and personable throughout the whole time, despite how crazily busy you always are.

I have been incredibly lucky to cross paths with some truly wonderful people at STOR-i and even more lucky to befriend them. I would like to thank my year group for all the good times we have spent together. I am also grateful for the friends I have made in other year groups. The list is long, but the following people cannot be omitted. Edwin

– thank you for being the cohesive glue in our squad, and for your hospitality during all the fun times we have had at yours. Georgia – thank you for being a tremendously caring friend. I cherish our numerous cathartic complaining sessions as much as all our culinary adventures. Livia – thank you for your support through all my driving related frustrations, and for the much needed catch-ups during the crazy times this year. Srshti – thank you for making me believe I am more qualified than I thought, for our joint endeavours to attempt Kaggle challenges, attend PyData events, and take on responsibilities at RSS. Thank you for helping me grow, and for growing with me. I would also like to thank my special friends and family outside STOR-i, 欢 and 英英, 所有和你们一起度过的平静又美好的时光都让我如沐春风！ Having both of you as my friends has been a true blessing in my life for the past few years. I cherish our countless hours of invigorating conversations during so many lovely home-made meals and pleasant walks. Thank you for making my days so much more bright and colourful than they would have been.

An extra special acknowledgement goes to Euan, for being the sunshine in my life. I cannot stop wondering how lucky I am to have you by my side. Your love and belief in me have made me a stronger person. Thank you for being my music curator, gig organiser, film recommender, game teacher, quiz master, and thesis proofreader. Thank you for reminding me how to have fun at times when I might have forgotten. Your sense of humour always has the magic power to lighten my heart. This journey would not have been anywhere near as good without your love and company, certainly not without all the delicious, comforting, and homely meals you have made.

My final acknowledgement of gratitude goes to my parents, without whom I would not have had the opportunity to undertake the path I have taken and become the person I am today. I am immensely grateful for my mother, for always instilling in me endless positive energy, and for constantly reminding me to be grateful for how much I have achieved thus far. I owe my gratitude to my father, for always pushing me to test my mettle and discipline myself; for teaching me how to persevere through difficult times, and how to stay humble when the days are rosy.

Declaration

I declare that the work in this thesis has been done by myself and has not been submitted elsewhere for the award of any other degree.

A version of Chapter 3 has been published as Peng, H., and Pavlidis, N. G. (2019). **Subspace Clustering with Active Learning.** In 2019 IEEE International Conference on Big Data (pages 135-144). IEEE.

A version of Chapter 5 has been published as Peng, H., Pavlidis, N. G., Eckley, I. A., and Tsalamanis, I. (2018). **Subspace Clustering of Very Sparse High-Dimensional Data.** In 2018 IEEE International Conference on Big Data (pages 3780-3783). IEEE.

Hankui Peng

Contents

Abstract **I**

Acknowledgements **III**

Declaration **V**

Contents **VI**

List of Figures **X**

List of Tables **XII**

1 Introduction **1**

1.1 Notation 2

1.2 Motivation 3

1.3 Thesis Contributions 5

2 Background **7**

2.1 K-Means Clustering 8

2.2 Graph Partitioning Problem 11

2.2.1 Similarity Graphs 11

2.2.2 Graph Cut Objectives 13

2.3 Spectral Clustering 14

2.3.1 Graph Laplacians 15

2.3.2 The Ratio Cut Problem 16

2.3.3 The Normalised Cut Problem 19

2.3.4 Spectral Clustering - An Example 23

2.4 Subspace Clustering 25
2.4.1 Iterative Methods 27
2.4.2 Spectral Methods 31
2.4.3 Algebraic Methods 35
2.4.4 Statistical Methods 43
2.5 Clustering Performance Measures 48
2.5.1 Purity & Clustering Error 49
2.5.2 Adjusted Rand Index (ARI) 50
2.5.3 Normalised Mutual Information (NMI) 52
2.A Appendix: Connection between Graph Cuts and Graph Laplacians . . . 54
2.A.1 The Ratio Cut and the Un-normalised Graph Laplacian 54
2.A.2 The Normalised Cut and the Normalised Graph Laplacians . . . 56

3 Subspace Clustering with Active Learning 59
3.1 Introduction 60
3.2 Related Work 61
3.3 Active Learning Framework 63
3.3.1 K-Subspace Clustering 63
3.3.2 Query Procedure 64
3.3.3 Update Procedure 71
3.4 Experimental Results 74
3.4.1 Synthetic Data 75
3.4.2 Real Data 77
3.5 Extension to Spectral Clustering 80
3.6 Conclusions & Future Work 83
3.A Appendix 84
3.A.1 Proof of Proposition 1 84
3.A.2 Proof of Proposition 2 85

4 Weighted Sparse Simplex Representation 87

4.1 Introduction 88
4.2 Literature Review 91
4.2.1 Subspace Clustering 91
4.2.2 Constrained Clustering 98
4.3 Weighted Sparse Simplex Representation (WSSR) 103
4.3.1 Sparse Simplex Representation (SSR) 103
4.3.2 Weighted Sparse Simplex Representation (WSSR) 104
4.4 Properties of WSSR 108
4.4.1 KKT Conditions for Optimality 109
4.4.2 Solving the Full WSSR Problem 110
4.4.3 Necessary Condition for the Trivial Solution 113
4.4.4 Sufficient Condition for the Trivial Solution 114
4.5 Constrained Clustering and Active Learning with WSSR (WSSR+) . . . 116
4.5.1 Constrained Clustering 117
4.5.2 Active Learning 118
4.6 Experiments on Synthetic Data 120
4.6.1 Varying Angles between Subspaces 121
4.6.2 Varying Noise Levels 122
4.6.3 Varying Subspace Dimensions 125
4.7 Experiments on Real Data 127
4.7.1 WSSR Experiments on MNIST Data 127
4.7.2 WSSR Experiments on USPS Data 132
4.7.3 WSSR+ Experiments 133
4.8 Conclusions & Future Work 136
4.A Appendix 138
4.A.1 WSSR+ Experiments on UCI Benchmark Data 138
4.A.2 WSSR+ Experiments on Cancer Gene Data 140

5 Clustering the Amazon Web-Scraped Text Data 143

5.1 Introduction 144

5.2 Vector Space Representation 146
5.2.1 Document Term Matrix (DTM) 147
5.2.2 Term Frequency - Inverse Document Frequency (TF-IDF) . . . 148
5.3 Document Clustering Methods 151
5.3.1 Principal Direction Divisive Partitioning (PDDP) 151
5.3.2 Latent Dirichlet Allocation (LDA) 153
5.4 Minimum Angle Clustering (MAC) 156
5.5 Experimental Results 160
5.6 Conclusions & Future Work 163

6 Conclusions & Future Work 165

Bibliography 169

List of Figures

1.2.1 A collection of points sampled in a three-dimensional ambient space from a union of three subspaces. 4

2.1.1 An example of applying K-means clustering to two data sets both with convex clusters. Also shown are the location updates of cluster centres for a total number of 10 iterations. **Left:** K-means successfully identifies three clusters that are of the same size. **Right:** K-means fails when the cluster sizes are very imbalanced. 9

2.1.2 Visualisation of K-means clustering results with varying K applied to two data sets with non-linearly separable clusters. 10

2.3.1 Histograms of the degree distribution based on the affinity matrix. . . . 25

2.3.2 A visualisation of the two eigenvectors corresponding to the two smallest eigenvalues of L_{sym} (first and second row), and the data points coloured in the assigned cluster labels (third row). 26

2.4.1 Data points drawn from a union of two subspaces in $\mathbb{R}^3$. 40

3.4.1 Performance results measured by NMI on six motion segmentation data sets with KSC initialisation. 79

3.4.2 Performance results measured by NMI on Yale Faces data sets with KSC initialisation. 82

4.3.1 An illustration of the effect of the data normalisation step, which provides the rationale for the use of the inverse cosine similarity. **Left:** The original data points. **Right:** The data points that have been normalised to lie on the unit sphere. 105

4.3.2 A geometric illustration of the necessity for stretching points in Y. . . 107
4.4.1 A geometric interpretation for when the trivial solution is obtained. . . 116
4.6.1 Three illustrations of data from two clusters under varying angles between the two. 121
4.6.2 Three illustrations of data from two clusters with varying noise levels. 123
4.6.3 Visualisation of the affinity matrix in the noise-free scenario. 124
4.6.4 Performance results of three methods across varying subspace dimensions. For each subspace dimension q, 20 trials are conducted with randomly chosen neighbourhood size k. 126
4.7.1 Median running times (in log-scale of seconds) of different algorithms on the MNIST handwritten digits data. 131
4.A.1 The clustering accuracy (min, median, max) of various constrained clustering algorithms over 20 trials. 140

5.2.1 Word clouds containing the most frequent words in the Amazon data. . 147
5.2.2 An illustration of the subspace structure of the US Amazon data. . . . 150
5.4.1 Histogram of the number of points in each subspace identified through the Reduced Row Echelon Form (RREF) of the TF-IDF matrix. 158

List of Tables

2.1 Notation for comparing two set of labels on the same data set. 51

3.1 The percentage of points queried before perfect cluster performance (as evaluated by NMI) is reached on synthetic data sets. 76

3.2 Cluster performance of various active learning strategies on motion segmentation data sets. 78

3.3 Cluster performance on Yale Faces data sets. 81

4.1 Accuracy of various subspace clustering algorithms on synthetic data with varying angles between subspaces. 122

4.2 Accuracy of various subspace clustering algorithms on synthetic data with varying noise levels. 123

4.3 Accuracy of various subspace clustering algorithms on synthetic data with varying subspace dimensions. 125

4.4 Median clustering accuracy along with the standard deviations on the MNIST handwritten digits data across 20 trials with $P = 200$. 129

4.5 Median clustering accuracy along with the standard deviations on the MNIST handwritten digits data across 20 trials with $P = 500$. 130

4.6 Median clustering accuracy along with the standard deviations on the USPS data across 20 trials. 132

4.7 Clustering accuracy of various constrained clustering methods on the MNIST data. The initial affinity matrix for all methods is produced by WSSR. 134

4.8 Clustering accuracy of various constrained clustering methods on the USPS data. The initial affinity matrix for all methods is produced by WSSR. 135
4.9 A summary of the UCI benchmark data sets. 138
4.10 Clustering accuracy of various spectral-based constrained clustering methods on UCI benchmark data sets. 139
4.11 Summary information on the gene expression data sets. 141
4.12 Performance comparison (using accuracy) with state-of-the-art (constrained) subspace clustering methods. 142

5.1 A summary of the five categories in the Amazon web-scraped data. . . . 146
5.2 A list of the ten most highly weighted words within each product category according to the TF-IDF representation. 149
5.3 A summary of notations for LDA. 154
5.4 Clustering performance and runtime comparison (in seconds) on the Amazon data set using TF-IDF representation. 162

Chapter 1

Introduction

The early history of cluster analysis dates back to Driver and Kroeber (1932) where it was first applied in anthropology. Clustering is the art of grouping a collection of unlabelled data points (usually represented as a vector of measurements in a multidimensional space) into a number of clusters, such that data points lie in the same cluster are more similar to each other compared to data points in different clusters (Jain et al., 1999). Myriad applications of clustering can be found across many fields, for example biological sequence analysis, medical imaging, social network analysis, and recommender systems (Guo, 2013).

Due to the growing computational capacity in recent years, many applications in the aforementioned fields are able to collect and process data in gigantic amounts and with a large number of features. Classical clustering methods such as K-means clustering (MacQueen, 1967) can still be applied to large-scale problems, whilst maintaining a similar level of cluster performance. However when the number of features is much larger than the number of data points, it becomes less straightforward and potentially ineffective to directly apply the existing clustering methodologies due to the *curse-of-dimensionality* (Bellman, 1966). It refers to the fact that the volume of space increases exponentially as the dimensionality increases, which means that the amount of data that can densely fill a low-dimensional space would become extremely sparse in higher dimensions. As a result, the Euclidean distances among all pairs of points become more and more similar to each other with the increase of dimensions.

As such, it is desirable to have methods that can handle high-dimensional data effectively and efficiently. A large amount of research has emerged in recent years to tackle the challenges of high-dimensionality in clustering, for example in gene sequencing (McWilliams and Montana, 2014), motion segmentation (Rao et al., 2010), and image recognition (You et al., 2016). It has been observed that high-dimensional data often lie in lower dimensional linear / affine subspaces or non-linear manifolds, rather than uniformly distributed in a high-dimensional ambient space (Elhamifar and Vidal, 2013).

Some previous work have approached problems in motion segmentation and image recognition with manifold clustering techniques (Saul and Roweis, 2003; Souvenir and Pless, 2005; Goh and Vidal, 2007; Elhamifar and Vidal, 2011). Many of these methods utilise the fact that points that lie in the same local neighbourhood of a manifold can be well approximately by a low-dimensional affine subspace (Saul and Roweis, 2003). As such, subspace properties can be used to obtain pairwise proximity between points and to ultimately obtain the data segmentation. The type of methods that model a collection of high-dimensional data as a union of lower dimensional subspaces is referred to as *subspace clustering* (Vidal, 2011), which is the main focus of this thesis.

1.1 Notation

We aim to use a consistent notation throughout this thesis. However, it means that our adopted notations may at times deviate from some of the conventions used in the related literature. Scalars are denoted by lowercase letters, such as $x \in \mathbb{R}$. Vectors are denoted by lowercase bold letters, such as $\boldsymbol{x} \in \mathbb{R}^P$. All vectors are assumed to be column vectors. Mathematical sets are denoted by uppercase calligraphic letters, such as $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^N$. Matrices are denoted by uppercase letters, such as $X = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\mathsf{T} \in \mathbb{R}^{N \times P}$. The i-th row of X is denoted as $X_{i\cdot}$ ($i \in \{1, \ldots, N\}$), and the j-th column of X is denoted as $X_{\cdot j}$ ($j \in \{1, \ldots, P\}$).

The P by P identity matrix is denoted by I_P, which is abbreviated to I when there is no ambiguity about its dimensionality. We denote $\mathbf{1}_P = \text{diag}\,(I_P) = [1, \ldots, 1]^\mathsf{T}$ as the P-dimensional vector with all ones, which corresponds to the vector that contains the diagonal entries of I_P. We abbreviate $\mathbf{1}_P$ to $\mathbf{1}$ when there is no ambiguity about its dimensionality. We use $\boldsymbol{e}_i$ to denote the basis vector with appropriate dimensionality, in which it takes the value one at the i-th location and zero everywhere else.

1.2 Motivation

An example of clustering data that contain groups of points from varying subspaces is shown in Figure 1.2.1. Some data points drawn from two one-dimensional subspaces $\mathcal{S}_1$ and $\mathcal{S}_2$, and other data points lie on a two-dimensional plane $\mathcal{S}_3$. In general, given a collection of data points $\left\{\boldsymbol{x}_i \in \mathbb{R}^P\right\}_{i=1}^N$ drawn from a union of K linear or affine subspaces $\{\mathcal{S}_k\}_{k=1}^K$ with dimensions $q_k = \dim(\mathcal{S}_k)$, $0 < q_k < P$, a subspace $\mathcal{S}_k$ can be defined as follows (Vidal, 2011)

$$\mathcal{S}_k = \left\{\boldsymbol{x} \in \mathbb{R}^P, \boldsymbol{x} = \boldsymbol{\mu}_k + V_k\boldsymbol{y}\right\}, \quad k \in \{1, \ldots, K\}. \tag{1.2.1}$$

In Eq. (1.2.1), $\boldsymbol{\mu}_k \in \mathbb{R}^P$ is an arbitrary point in $\mathcal{S}_k$ that is chosen as $\boldsymbol{\mu}_k = \mathbf{0}$ for linear subspaces. The columns of $V_k \in \mathbb{R}^{P\times q_k}$ are the orthonormal basis vectors for subspace $\mathcal{S}_k$ which need not to be unique, and $\boldsymbol{y} \in \mathbb{R}^{q_k}$ is the low-dimensional representation of $\boldsymbol{x}$. The goal of subspace clustering is to find the number of subspaces K, the displacements $\{\boldsymbol{\mu}_k\}_{k=1}^K$, their subspace dimensions $\{q_k\}_{k=1}^K$ and bases $\{V_k\}_{k=1}^K$, along with the partition of points according to the subspaces.

Two common dependence structures between subspaces are *independent* and *disjoint* subspaces, which we provide the definitions for as follows (Soltanolkotabi and Candes, 2012).

Definition 1.2.1. A collection of subspaces $\{\mathcal{S}_k\}_{k=1}^K$ is said to be *independent* if the dimension of the union of subspaces is equal to the sum of the subspace dimensions, i.e. $\dim(\oplus_{k=1}^K \mathcal{S}_k) = \sum_{k=1}^K \dim(\mathcal{S}_k)$, where $\oplus$ denotes the direct sum operator.

Definition 1.2.2. A collection of subspaces $\{\mathcal{S}_k\}_{k=1}^K$ is said to be *disjoint* if every pair of subspaces $\mathcal{S}_i$ and $\mathcal{S}_j$ intersect only at the origin, i.e. $\dim(\mathcal{S}_i \oplus \mathcal{S}_j) = \dim(\mathcal{S}_i) + \dim(\mathcal{S}_j), \quad \forall i, j \in \{1, \ldots, K\}$.

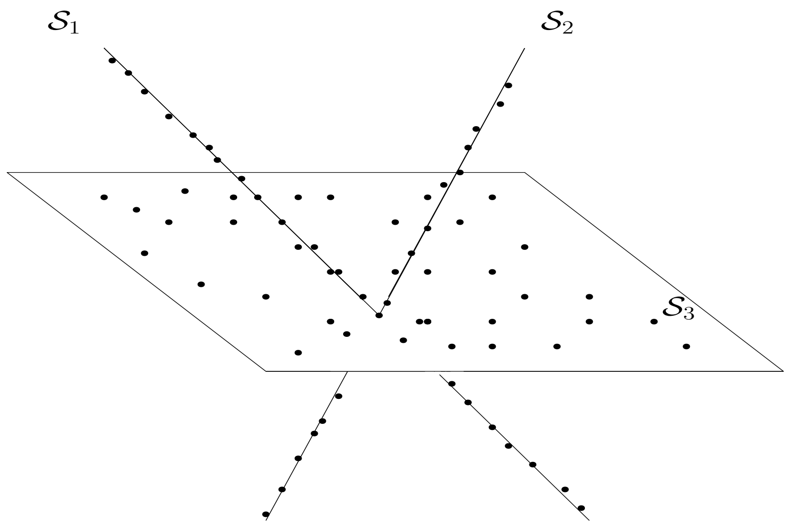


Figure 1.2.1: A collection of points sampled in a three-dimensional ambient space from a union of three subspaces.

When $K = 1$, the above quantities can be obtained through Principal Component Analysis (PCA) (Jolliffe, 2011). The problem reduces to one of finding a few principal components that can capture most of the variability in the data. However when $K > 1$, there are a number of challenges that make the subspace clustering problem difficult to solve. Below are some of the well-known challenges (Vidal, 2011):

- It is often difficult to choose or design an appropriate measure of similarity / distance among the high-dimensional data points.
- The existence of noise and potentially outliers in the data from many real world applications require the development of robust subspace estimation techniques.
- The position and orientation of different subspaces can be arbitrary, and the existence of dependence structure between subspaces makes the problem more difficult to solve.

On top of all the aforementioned issues, a general challenge in clustering real world data lies in the difficulty of validating the cluster performance due to the scarcity of labelling information. In practice, it is often feasible to obtain some form of external information either as labels, or in the form of 'must-link' and 'cannot-link' constraints which indicate whether pairs of points belong to the same cluster or not. The model performance could then be improved by both satisfying the constraints imposed by the external information, and taking advantage of the external information on the labelled points to improve the cluster performance on the unlabelled points. The problem of clustering while utilising a fixed amount of labelling information is called *constrained clustering* (Basu et al., 2008).

However, randomly labelling a small amount of the data would not necessarily guarantee that the cluster performance would improve the most if at all (Wagstaff, 2006). This is because the information contained in partial labels or pairwise constraint set does not necessarily get propagated to the unlabelled points. Setting the pairwise similarity to zero for points that are known to have 'cannot-link' relationships does not mean that they will definitely get assigned to different clusters (Li et al., 2009). Therefore, it is desirable to query the external information in an active manner, so that the cluster performance would improve effectively and efficiently over time. The problem of iteratively querying informative and potentially misclassified data so as to maximally improve the model performance is known as *active learning* (Settles, 2009).

1.3 Thesis Contributions

Our research creates a unified framework for subspace clustering, constrained clustering, and active learning. This thesis contains three main methodological contributions.

In Chapter 3, we propose a **Subspace Clustering with Active Learning (SCAL)** framework (Peng and Pavlidis, 2019) for the K-subspace clustering (KSC) algorithm (Agarwal and Mustafa, 2004). KSC is a K-means-like iterative algorithm that alternates between subspace estimation and cluster assignment. Although the algorithm usually converges in a few iterations, it is only guaranteed to converge to a local optimum. Our proposed framework consists of two stages that sequentially improve the performance of KSC in an effective and efficient manner.

In the first stage, our proposed active learning strategy exploits the structure of the current subspaces and queries the most informative and potentially misclassified points. In the second stage, we propose a constrained subspace clustering algorithm which updates the cluster labels and subspace structure based on the queried information whilst satisfying the constraints imposed by the queried points. The proposed framework is designed for iterative subspace clustering methods. However, it can also be applied to other types of subspace clustering methods, for example, spectral-based subspace clustering.

In Chapter 4, we design a unified framework of active learning and constrained clustering for spectral-based subspace clustering methods. We propose a spectral-based subspace clustering methodology, named **Weighted Sparse Simplex Representation (WSSR)**. It has been shown to enjoy excellent performance in a range of synthetic and real data sets. In the presence of a fixed amount of labelling information or pairwise constraints, we show that our proposed methodology is flexible enough to incorporate them into the problem formulation and satisfy the constraints, thus leading to effective improvement in the cluster performance. Finally we show that our proposed active learning strategy in Chapter 3 can be naturally incorporated in the spectral-based setting.

In Chapter 5, we develop an algebraic subspace clustering methodology named **Minimum Angle Clustering (MAC)** (Peng et al., 2018). It is motivated by the application of clustering Amazon product names, which are mostly composed of very short texts. The resulting TF-IDF representation for the text data are both sparse and high-dimensional. However, most of the variability for each category can be captured well with a much lower dimensional subspace. MAC first utilises the Reduced Row Echelon Form (RREF) technique to identify a large number of subspaces that each contains very few points. We propose a subspace proximity measure based on principal angles (Drmac, 2000), which is used to merge the large number of subspaces into meaningful clusters. On the application of clustering Amazon product names, MAC has been shown to perform favourably against other well-established document clustering and subspace clustering methods.

Chapter 2

Background

In this chapter, we first provide a review of the two most fundamental algorithms in clustering, which are the building blocks of our work in later chapters. In Section 2.1, we introduce K-means clustering algorithm (Forgy, 1965; MacQueen, 1967), which groups data points into a pre-specified number of clusters by assigning each data point to its closest centroid in an iterative manner. However, K-means clustering can only identify clusters with spherical shapes. One approach to overcome this limitation is by first building a similarity graph of the data, and then solving a graph partitioning problem. In Section 2.2, we discuss how to build a similarity graph of the data before introducing two types of graph partitioning problems. These problems are NP-hard to solve, but their relaxations can be solved via the eigen-decomposition of the graph Laplacian matrices. We introduce two popular spectral clustering algorithms in Section 2.3, and discuss their connections to graph cut problems. In Section 2.4, we review the relevant literature for each of the four main categories of subspace clustering methods: algebraic, iterative, spectral, and statistical methods. Finally, we introduce the most commonly used external performance measures for clustering in Section 2.5 – Purity (Zhao and Karypis, 2001), Adjusted Rand Index (ARI) (Hubert and Arabie, 1985), and Normalised Mutual Information (NMI) (Cover and Thomas, 2012). When the ground truth labels are available, these measures can be used to evaluate the agreement between the assigned cluster labels and the ground truth labels.

2.1 K-Means Clustering

K-means clustering is one of the most fundamental and well-known algorithms in clustering. The term 'K-means' is first used in MacQueen (1967), though it is also known as Lloyd-Forgy since Forgy proposed essentially the same method (Forgy, 1965). It groups data into a predefined number of clusters such that the points that lie in the same cluster are closer to each other, most commonly in terms of the Euclidean distance, as compared to points in different clusters. It is an iterative algorithm that alternates between: (a) calculating the cluster centres given the cluster labels; and (b) updating the cluster labels given the cluster centres. Given a data set $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ with N points, K-means clustering minimises the following objective function (See Section 9.1 in Bishop (2006)):

$$L(\mathcal{X}, \Omega) = \frac{1}{N} \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in \Omega_k} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|_2^2, \tag{2.1.1}$$

where $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\}$ denotes the set of K cluster centres and $\Omega = \{\Omega_1, \ldots, \Omega_K\}$ denotes a partitioning of the data into K clusters. The aim is to minimise the sum of squared Euclidean distances between all data points and their corresponding cluster centres.

Finding a global minimum to the objective in Eq. (2.1.1) is NP-hard (Aloise et al., 2009). K-means clustering is the most common algorithm to minimise the objective through iterative refinement. The algorithm in procedural form is stated as follows:

1. Given the number of clusters K, randomly select K distinct points as the initial cluster centres $\boldsymbol{\mu}_1^{(0)}, \ldots, \boldsymbol{\mu}_K^{(0)}$. [1]

2. **Assignment step:** For each data point $\boldsymbol{x}_i$, calculate the distance from $\boldsymbol{x}_i$ to all cluster centres. Assign $\boldsymbol{x}_i$ to the cluster whose centre it is closest to

$$\omega_i^{(t)} = \underset{k \in \{1,\ldots,K\}}{\arg\min} \left\| \boldsymbol{x}_i - \boldsymbol{\mu}_k^{(t-1)} \right\|_2^2, \quad \forall\, i \in \{1, \ldots, N\}, \tag{2.1.2}$$

where t ($t = 1, 2, 3, \ldots$) is the iteration number.

[1] K-means clustering is very sensitive to initialisation. Different initialisations can lead to very different results.

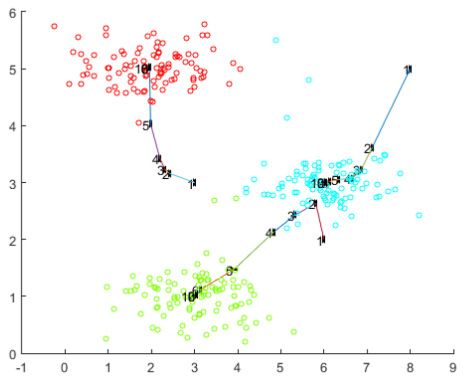
3. **Update step:** Recalculate the cluster centres by averaging over all data points that lie in the same cluster

$$\boldsymbol{\mu}_k^{(t)} = \frac{1}{n_k^{(t-1)}} \sum_{\boldsymbol{x} \in \Omega_k^{(t-1)}} \boldsymbol{x}, \tag{2.1.3}$$

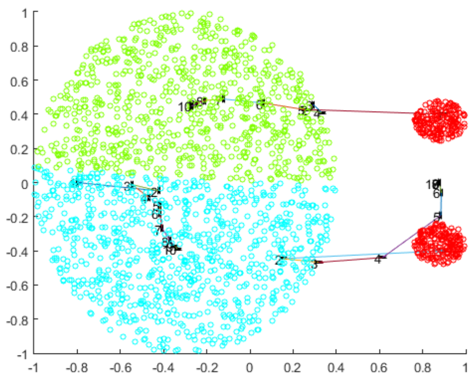
where $n_k^{(t-1)} = |\Omega_k^{(t-1)}|$ denotes the cardinality of cluster k in iteration $(t-1)$.

4. Iterate between step 2 and 3 until a stopping criterion is reached. The standard criterion is to stop if there is no further change to the cluster labels.

Although K-means clustering monotonically decreases the objective in Eq. (2.1.1), it is only guaranteed to converge to a local minimum.



(a) Balanced clusters.



(b) Imbalanced clusters.

Figure 2.1.1: An example of applying K-means clustering to two data sets both with convex clusters. Also shown are the location updates of cluster centres for a total number of 10 iterations. **Left:** K-means successfully identifies three clusters that are of the same size. **Right:** K-means fails when the cluster sizes are very imbalanced.

Figure 2.1.1 provides a visualisation of how K-means clustering updates the cluster centres over iterations. Each figure shows the path of each cluster centre at every iteration for a total number of 10 iterations for both data sets. The initial cluster centres of the data example in (a) are randomly initialised, whereas the initial cluster centres of the data example in (b) are chosen such that each cluster centre lies within their true class. We see that K-means clustering fails to recover the correct cluster labels in data example (b), in which one cluster is ostensibly larger than the other two. It is generally the case that K-means algorithm tends to generate similar-sized clusters.

In addition, K-means clustering is very sensitive to the locations of the initial cluster centres. Hence, either multiple initial cluster centres should be used or a sensible initialisation strategy should be adopted. There are numerous advanced initialisation strategies. For example, K-means++ (Arthur and Vassilvitskii, 2006) picks points as cluster centres in a sequential manner, which takes into account the Euclidean distance between each point to all of the existing cluster centres. The main idea is that the initial centroids should be far away from each other. It has been shown that this strategy improves both the speed and accuracy of K-means clustering.

Another drawback of the classical K-means clustering is that it cannot handle non-convex clusters. Shown in Figure 2.1.2 are the results of applying K-means to two data sets with non-convex clusters. Since none of these clusters is linearly separable, K-means algorithm struggles to recover the correct grouping regardless of the chosen number of clusters K.

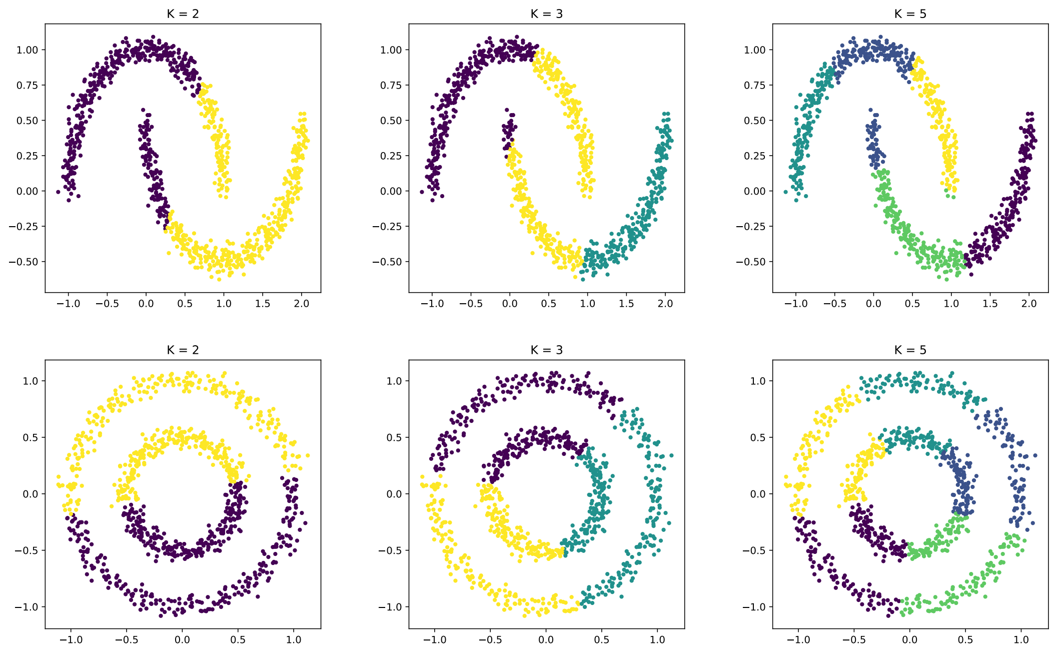


Figure 2.1.2: Visualisation of K-means clustering results with varying K applied to two data sets with non-linearly separable clusters.

In order to handle non-linearly separable clusters, one could use kernel K-means clustering (Girolami, 2002; Dhillon et al., 2004; Filippone et al., 2008). It maps the data to a higher-dimensional inner product feature space. The data vector in K-means is replaced with the projected data vector in kernel K-means. The distance from the projected data to their corresponding cluster centroids can be calculated through the use of kernel functions.

Another way of handling non-linearly separable clusters is by first building a suitable similarity graph / matrix of the data, then solving a graph partitioning problem to obtain the final cluster labels. This type of approach is called Spectral Clustering (Shi and Malik, 2000; Ng et al., 2002). Dhillon et al. (2004) have shown that a generalisation of the kernel K-means algorithm is equivalent to the normalised cut spectral clustering algorithm proposed in Ng et al. (2002). In the next two sections, we will familiarise the reader with graph partitioning problems, and provide a detailed introduction to spectral clustering.

2.2 Graph Partitioning Problem

A similarity graph uses nodes and edges to conceptually represent data points and the pairwise similarities between them. The problem of partitioning a graph mimics that of clustering data points into groups. In clustering, the aim is to keep points that are similar in the same group and points that are dissimilar in different groups. In graph partitioning, the aim is to partition a graph such that the resulting sub-graphs are well-connected by edges with high weights indicating high similarity between corresponding nodes. At the same time, the edges between different sub-graphs should have low weights. In this section, we first introduce different similarity graphs before introducing the graph partitioning problem. A thorough discussion on this topic and spectral clustering can be found in Von Luxburg (2007).

2.2.1 Similarity Graphs

Given a set of N data points $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and the pairwise similarity values $w_{ij} \in \mathbb{R}^+$, we can represent a data set and the connectivity information among the points through a similarity graph. A graph $G = (\mathcal{V}, \mathcal{E})$ is composed of a set of nodes

$\mathcal{V}$ and edges $\mathcal{E}$. Each node i corresponds to a data point $\boldsymbol{x}_i$, and each edge represents a connection between two data points. There are two types of graphs: directed and undirected. Each edge in a directed graph is pointed towards a node, and the two edge weights w_{ij} and w_{ji} between node i and j are not necessarily the same. Here we restrict our attention to undirected graphs, with $w_{ij} = w_{ji}$ for all pairs of i, j. The edge weights correspond to the pairwise similarity values between data points, with a zero edge weight indicating no connection. By default, we consider there is no connection between $\boldsymbol{x}_i$ and itself, i.e. $w_{ii} = 0$.

We introduce a few notions here to better characterise a similarity graph. An *adjacency matrix* A is an $N \times N$ matrix in which A_{ij} denotes the edge weight / similarity w_{ij} between node i and j. The connectivity of a node $\boldsymbol{x}_i$ is formally called the *degree*, d_i, which is calculated as the sum of all edge weights attached to the node, $d_i = \sum_{j=1}^{N} w_{ij}$. A *degree matrix* D is an $N \times N$ diagonal matrix, in which the i-th diagonal entry D_{ii} represents the degree of node i. Given a subset of vertices $\mathcal{S} \subset \mathcal{V}$, the complement of the subset is denoted as $\bar{\mathcal{S}} = \mathcal{V} \backslash \mathcal{S}$.

A subset $\mathcal{S}$ of a graph is *connected* if any pair of nodes in $\mathcal{S}$ can be connected by a sequence of edges whose corresponding nodes also lie in the set $\mathcal{S}$. In graph theory, a *connected component* of an undirected graph is a sub-graph in which there exists a path between any pair of nodes that are connected by a sequence of edges. Additionally, this sub-graph is connected to no additional nodes in the graph (Chung, 1997). For the purpose of clustering, we first introduce a few ways of constructing a similarity graph before discussing how to solve a graph cut problem.

The ε neighbourhood graph. This type of graph is constructed in such a way that two nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are connected if the distance between them is less than a certain threshold ε. Similarly, there exists an edge between two nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ if the pairwise similarity is above a certain threshold ε. In the ε neighbourhood graph, the pairwise relationship between nodes are either connected or not connected, thus it is usually considered as a type of unweighted graph.

The k nearest neighbour (k-NN) graph. The k-NN graph creates an edge between each node to its k nearest neighbours. In particular, we call it a *mutual k nearest neighbour graph* if an edge exists only if both nodes are in each other's k nearest neighbourhood. Another version of this graph is to allow an edge between a pair of data points as long as one of the nodes is in the other's k nearest neighbourhood. In both cases, a symmetric similarity matrix can be obtained. The k-NN graph is one of the most commonly used graphs in spectral clustering.

The fully connected graph. All pairwise edge weights are non-zero in this type of similarity graph. The edge weight is calculated either using a similarity function or distance measure. A common choice for a similarity function is the Gaussian similarity function: $w_{ij} = \exp\left\{\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2\sigma^2}\right\}$, in which σ is called the bandwidth parameter that controls the size of the neighbourhood. The choice of σ is crucial to the quality of the resulting partitioning of the graph. We can consider the ε neighbourhood graph as a pruned version of the fully connected graph, as it can be obtained from the fully connected graph after a threshold level ε is specified.

2.2.2 Graph Cut Objectives

Once a similarity graph is constructed, the next problem to be addressed is how to partition the graph into a number of sub-graphs. Ideally, one would want to cut through a small number of edges with low weights in order to obtain a well-connected and balanced partition. Well-connected in the sense that edges within each sub-graph should have relatively high weights, and balanced in the sense that the sizes of different sub-graphs are not too different from each other. Both of these are desirable properties in many real-world applications. For example, parallel computing involves the problem of assigning and processes evenly across processors whilst minimising communication (Andreev and Racke, 2006).

This leads one to ask the following two questions: (a) How many edges should we cut? (b) Which edges should we cut? To begin with, the *cut* for a K-partitioning on a set of nodes $\mathcal{S}$ is defined as

$$\mathrm{cut}(\mathcal{S}_1, \dots, \mathcal{S}_K) := \frac{1}{2}\sum_{k=1}^{K} W(\mathcal{S}_k, \bar{\mathcal{S}}_k), \tag{2.2.1}$$

where $W(\mathcal{S}_k, \bar{\mathcal{S}}_k) := \sum_{\boldsymbol{x}_i \in \mathcal{S}_k, \boldsymbol{x}_j \in \bar{\mathcal{S}}_k} w_{ij}$. Note that this is simply the sum of weights for all edges that need to be cut in order to obtain the partition. It does not take into account the sparsity of the cut, in that two different graph partitions that cut through different number of edges could achieve the same value according to Eq. (2.2.1). Furthermore, it also does not consider whether the sizes of the sub-graphs are similar or not.

We introduce two common graph cut objectives that include these two criteria: *ratio cut* (Hagen and Kahng, 1992) and *normalised cut* (Shi and Malik, 2000). The main difference between the two lies in how the size of a set is measured. Ratio cut measures the size of a set $\mathcal{S}$ through its cardinality, $|\mathcal{S}|$, and normalised cut measures the size of a set through the total edge weights contained in a set, $\mathrm{vol}(\mathcal{S}) = \sum_{i \in \mathcal{S}} d_i$. Explicitly, these two criteria can be expressed as follows

$$\mathrm{RatioCut}(\mathcal{S}_1, \dots, \mathcal{S}_K) := \frac{1}{2}\sum_{k=1}^{K} \frac{W(\mathcal{S}_k, \bar{\mathcal{S}}_k)}{|\mathcal{S}_k|} = \sum_{k=1}^{K} \frac{\mathrm{cut}(\mathcal{S}_k, \bar{\mathcal{S}}_k)}{|\mathcal{S}_k|}. \tag{2.2.2}$$

$$\mathrm{NCut}(\mathcal{S}_1, \dots, \mathcal{S}_K) := \frac{1}{2}\sum_{k=1}^{K} \frac{W(\mathcal{S}_k, \bar{\mathcal{S}}_k)}{\mathrm{vol}(\mathcal{S}_k)} = \sum_{k=1}^{K} \frac{\mathrm{cut}(\mathcal{S}_k, \bar{\mathcal{S}}_k)}{\mathrm{vol}(\mathcal{S}_k)}. \tag{2.2.3}$$

Both of these criteria can be optimised through minimising $\mathrm{cut}(\mathcal{S}_i, \bar{\mathcal{S}}_i)$ and maximising the size of each subset simultaneously. Algorithmically, these objectives that incorporate both considerations are NP-hard to solve (Wagner and Wagner, 1993). One heuristic approach that solves a relaxation of the graph cut problem guided by these two criteria is spectral clustering, which we will introduce next in Section 2.3.

2.3 Spectral Clustering

In this section, we introduce different spectral clustering algorithms that solve a relaxed version of the graph cut problem as discussed in Section 2.2. A relaxation of the graph cut problems can be solved through the eigen-decomposition of a graph Laplacian matrix, which can be obtained from a similarity graph. Ideas are borrowed from spectral graph theory (Chung, 1997) to circumvent the complexity of directly optimising the graph cut objectives.

We first discuss the most common forms of graph Laplacians and their properties in Section 2.3.1. We demonstrate the connection between spectral clustering and the graph cut problem with ratio cut and normalised cut objectives in Section 2.3.2 and Section 2.3.3. A working example is provided in Section 2.3.4 to illustrate the mechanism of spectral clustering on a synthetic data set.

2.3.1 Graph Laplacians

A graph Laplacian matrix contains information about the connectivity within a graph. Spectral graph theory (Chung, 1997) is a field that studies the properties of different graph Laplacian matrices.

Un-normalised graph Laplacian. There are different forms of graph Laplacian matrices. The most simple un-normalised graph Laplacian L is defined as follows (Cvetković et al., 1980)

$$L = D - A, \tag{2.3.1}$$

where $D \in \mathbb{R}^{N \times N}$ is the degree matrix and $A \in \mathbb{R}^{N \times N}$ is the weighted adjacency matrix, as previously introduced in Section 2.2.1. The weighted adjacency matrix A can also be called the *affinity matrix*. We use these two terms interchangeably in this thesis.

Many properties of the un-normalised graph Laplacian provide useful insights into the graph partitioning problem. For example in a bi-partitioning problem, the data are well separated when represented using the eigenvector corresponding to the second smallest eigenvalue of the graph Laplacian matrix. In a graph partitioning problem with K connected components (clusters), the data points (nodes) represented using the eigenvectors corresponding to the K zero eigenvalues of the graph Laplacian matrix are well separated in the K-dimensional eigen space. As such, the partitioning can be trivially detected through a simple clustering algorithm such as the K-means clustering.

We state the relevant properties here that will be useful for our illustration of spectral clustering later. Firstly, the un-normalised graph Laplacian is a symmetric positive semi-definite matrix, and its eigenvalues satisfy $0 = \lambda_1 \leqslant \lambda_2 \leqslant \ldots \leqslant \lambda_N$. Secondly, the smallest eigenvalue is always zero and the corresponding eigenvector is $\mathbf{1}$.

Normalised graph Laplacian. There are two well-known forms of the normalised graph Laplacian matrix (Chung, 1997). The first one is a symmetric matrix, which is defined as

$$L_{\text{sym}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}. \tag{2.3.2}$$

The second one is closely related to a random walk on a graph, which is defined as

$$L_{\text{rw}} = D^{-1} L = I - D^{-1} A. \tag{2.3.3}$$

A random walk on a graph is a stochastic process which jumps from node to node. The transition probability of jumping from node i to j can be expressed in terms of the edge weight w_{ij} as $p_{ij} = \frac{w_{ij}}{d_i}$. The transition matrix P can thus be expressed as $P = D^{-1}A$, which is equivalent to $I - L_{\text{rw}}$.

Many properties of the normalised graph Laplacian matrix share with the properties of the un-normalised version. For example, zero is an eigenvalue of both forms of the graph Laplacian matrix with the constant one eigenvector $\mathbf{1}$ up to a multiplying constant. In addition, both forms of the normalised graph Laplacian matrix are positive semi-definite, and have eigenvalues $0 = \lambda_1 \leqslant \lambda_2 \leqslant \ldots \leqslant \lambda_N$.

2.3.2 The Ratio Cut Problem

Previously, we introduced the ratio cut objective in Section 2.2.2 Eq. (2.2.2). In this section, we first restate the objective in terms of the un-normalised graph Laplacian matrix L. In the case of bi-partitioning and more generally K-partitioning, we show how a relaxation of the ratio cut problem can be solved, and demonstrate the equivalence of this relaxation to un-normalised spectral clustering (Von Luxburg, 2007).

The aim of the ratio cut problem in the bi-partitioning setting is to find two subsets

$\{\mathcal{S}_1, \mathcal{S}_2\}$ that minimises

$$\text{RatioCut}\left(\mathcal{S}_1, \mathcal{S}_2\right) = \frac{\text{cut}(\mathcal{S}_1, \mathcal{S}_2)}{|\mathcal{S}_1|} + \frac{\text{cut}(\mathcal{S}_2, \mathcal{S}_1)}{|\mathcal{S}_2|}, \tag{2.3.4}$$

such that $\mathcal{S}_1 \cup \mathcal{S}_2 = \mathcal{S}$ and $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$. Let $\boldsymbol{f} = [f_1, ..., f_N]^\mathsf{T}$ be an indicator vector defined as

$$f_i = \begin{cases} \sqrt{\frac{|\mathcal{S}_2|}{|\mathcal{S}_1|}}, & \boldsymbol{x}_i \in \mathcal{S}_1, \\ -\sqrt{\frac{|\mathcal{S}_1|}{|\mathcal{S}_2|}}, & \boldsymbol{x}_i \in \mathcal{S}_2, \end{cases} \tag{2.3.5}$$

for $i \in \{1, \ldots, N\}$. One can show that minimising the ratio cut objective in (2.3.4) is equivalent to minimising $\boldsymbol{f}^\mathsf{T} L \boldsymbol{f}$ subject to some constraints, which can be expressed as a discrete minimisation problem as follows

$$\begin{aligned} \min_{\mathcal{S}_1, \mathcal{S}_2 \subset \mathcal{S}} \quad & \boldsymbol{f}^\mathsf{T} L \boldsymbol{f} \\ \text{s.t.} \quad & \boldsymbol{f} \perp \mathbf{1}, \\ & \|\boldsymbol{f}\|_2 = \sqrt{N}, \\ & f_i \text{ as defined in (2.3.5)}, \\ & \mathcal{S}_1 \cup \mathcal{S}_2 = \mathcal{S}, \quad \mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset. \end{aligned} \tag{2.3.6}$$

The first two constraints follow from the definition of $\boldsymbol{f}$. A deduction for this equivalence can be found in Appendix 2.A.1.

The discreteness in the entries of $\boldsymbol{f}$ makes the problem in Eq. (2.3.6) NP-hard to solve (Wagner and Wagner, 1993). One obvious relaxation of this problem is to allow the entries in $\boldsymbol{f}$ to take arbitrary values in $\mathbb{R}$. Recall that the smallest eigenvalue is zero that corresponds to the eigenvector with all ones, $\mathbf{1}$. The solution of this relaxed problem $\boldsymbol{f}^\star$ is given by the eigenvector that corresponds to the second smallest eigenvalue of L, which is orthogonal to $\mathbf{1}$ (Lütkepohl, 1996). Thus, $(\boldsymbol{f}^\star)^\mathsf{T} L \boldsymbol{f}^\star$ serves as an approximate minimiser to the problem in Eq. (2.3.6).

In order to obtain a bi-partitioning of the graph G, we need to transform the real-valued entries in $\boldsymbol{f}^\star$ back to discrete-valued indicators. We can consider the entries in $\boldsymbol{f}^\star$

as points in $\mathbb{R}$, and apply K-means clustering to $\boldsymbol{f}^\star$ to obtain two clusters. This is exactly the procedure for the un-normalised spectral clustering algorithm in the case of $K = 2$.

For a general K-partitioning problem, we can re-express the ratio cut objective in terms of the un-normalised graph Laplacian L in the same vein as in the bi-partitioning scenario. Given a partition of $\mathcal{S}$ into K sets $\{\mathcal{S}_1, \ldots, \mathcal{S}_K\}$, the ratio cut objective can be expressed as

$$\text{RatioCut}(\mathcal{S}_1, ..., \mathcal{S}_K) = \sum_{k=1}^{K} \frac{\text{cut}(\mathcal{S}_k, \bar{\mathcal{S}}_k)}{|\mathcal{S}_k|}.$$

We denote $H \in \mathbb{R}^{N \times K}$ as an indicator matrix in which

$$H_{ik} = \begin{cases} \frac{1}{\sqrt{|\mathcal{S}_k|}} & \boldsymbol{x}_i \in \mathcal{S}_k, \\ 0 & \text{otherwise}, \end{cases} \tag{2.3.7}$$

for $i \in \{1, \ldots, N\}$ and $k \in \{1, \ldots, K\}$. Let $\boldsymbol{h}_k \in \mathbb{R}^N$ be the k-th column in H, then we have $H = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_K]$. One can show that the following holds

$$\frac{\text{cut}\left(\mathcal{S}_k, \bar{\mathcal{S}}_k\right)}{|\mathcal{S}_k|} = (\boldsymbol{h}_k)^\mathsf{T} L \boldsymbol{h}_k = \left(H^\mathsf{T} L H\right)_{kk}, \quad \forall\, k \in \{1, \ldots, K\}. \tag{2.3.8}$$

A deduction for this equivalence can be found in Appendix 2.A.1. As such, we have that

$$\text{RatioCut}(\mathcal{S}_1, ..., \mathcal{S}_K) = \sum_{k=1}^{K} \left(H^\mathsf{T} L H\right)_{kk} = \text{tr}\left(H^\mathsf{T} L H\right). \tag{2.3.9}$$

Therefore, we have transformed the ratio cut objective for general K into the following discrete trace minimisation problem involving the un-normalised graph Laplacian:

$$\begin{aligned} \min_{\mathcal{S}_1, \ldots, \mathcal{S}_K \subset \mathcal{S}} \quad & \text{tr}(H^\mathsf{T} L H) \\ \text{s.t.} \quad & H^\mathsf{T} H = I, \\ & H \text{ as defined in (2.3.7)}, \\ & \bigcup_{k=1}^{K} \mathcal{S}_k = \mathcal{S}, \\ & \mathcal{S}_i \cap \mathcal{S}_j = \emptyset, \quad \forall i, j \in \{1, \ldots, K\}. \end{aligned} \tag{2.3.10}$$

Again, we can relax the problem by allowing the entries in H to take arbitrary values in $\mathbb{R}$. According to the Rayleigh-Ritz theorem (Lütkepohl, 1996), the solution of the relaxed problem is given by $H^{\star}$, whose columns are the K eigenvectors of L that correspond to its K smallest eigenvalues. Since the entries in $H^{\star}$ are continuous approximations of H which encodes the exact partitioning information by construction, we can obtain the final partitioning by applying K-means clustering to the rows of $H^{\star}$ instead. This is the procedure for the un-normalised spectral clustering algorithm for general K. An algorithmic form for un-normalised spectral clustering is shown in Algorithm 1.

```
Algorithm 1: Un-normalised Spectral Clustering
  Input  : Data affinity matrix $A \in \mathbb{R}^{N \times N}$
           Number of clusters $K$

    1. Compute the un-normalised graph Laplacian: $L = D - A$
    2. Compute the eigen-decomposition of $L$
    3. Let $V \in \mathbb{R}^{N \times K}$ be the matrix whose columns contain the eigenvectors
       $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K$ corresponding to the $K$ smallest eigenvalues
    4. Apply $K$-means clustering to the rows of $V$ to obtain the final cluster labels
       $\Omega = \{\omega_1, \ldots, \omega_N\}$

  Output : Clusters $\{\mathcal{S}_1, \ldots, \mathcal{S}_K\}$ with $\mathcal{S}_k = \{i | \omega_i = k\}$ for $k \in \{1, \ldots, K\}$
```

It is worth pointing out that there is no guarantee on how close the solution obtained from spectral clustering is to that of the optimal solution of the ratio cut objective. In addition, the aforementioned relaxation approach is not unique. The popularity of this relaxation approach is mainly due to the simplicity in the resulting linear algebra problem (Von Luxburg, 2007).

2.3.3 The Normalised Cut Problem

In the previous section, we have demonstrated the connection between the ratio cut problem and the un-normalised spectral clustering algorithm. In this section, we further discuss the connection between the normalised cut problem (see Section 2.2.2) and two well-known normalised spectral clustering algorithms (Shi and Malik, 2000; Ng et al., 2002). Both of these spectral clustering algorithms solve an approximation of the normalised cut problem involving the use of different normalised graph Laplacians, which we previously introduced in Section 2.3.1.

The normalised spectral clustering algorithm proposed in Shi and Malik (2000) uses the random walk graph Laplacian matrix L_{rw}. We refer to this algorithm as the *random walk spectral clustering algorithm*. The other normalised spectral clustering algorithm proposed in Ng et al. (2002) uses the symmetric graph Laplacian matrix L_{sym}. We refer to this version as the *symmetric spectral clustering algorithm*.

We first show how the normalised cut problem can be re-expressed as a discrete optimisation problem involving the un-normalised Laplacian matrix L. Through change of variables, we transform the optimisation problem into two different formulations involving L_{rw} and L_{sym} respectively. We show that the relaxations of these two formulations lead to the two different normalised spectral clustering algorithms.

In the bi-partitioning setting, the normalised cut objective can be expressed as follows

$$\text{NCut}\left(\mathcal{S}_1, \mathcal{S}_2\right) = \frac{\text{cut}(\mathcal{S}_1, \mathcal{S}_2)}{\text{vol}\left(\mathcal{S}_1\right)} + \frac{\text{cut}(\mathcal{S}_2, \mathcal{S}_1)}{\text{vol}\left(\mathcal{S}_2\right)}, \tag{2.3.11}$$

such that $\mathcal{S}_1 \cup \mathcal{S}_2 = \mathcal{S}$ and $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$. The entries in the indicator vector $\boldsymbol{f}$ are defined as

$$f_i = \begin{cases} \sqrt{\frac{\text{vol}(\mathcal{S}_2)}{\text{vol}(\mathcal{S}_1)}}, & \boldsymbol{x}_i \in \mathcal{S}_1, \\ -\sqrt{\frac{\text{vol}(\mathcal{S}_1)}{\text{vol}(\mathcal{S}_2)}}, & \boldsymbol{x}_i \in \mathcal{S}_2. \end{cases} \tag{2.3.12}$$

for $i \in \{1, \ldots, N\}$. Similar to the ratio cut scenario, one can show that $\boldsymbol{f}^{\mathsf{T}} L \boldsymbol{f} = \text{vol}\left(\mathcal{S}\right) \cdot \text{NCut}(\mathcal{S}_1, \mathcal{S}_2)$, $\left(D\boldsymbol{f}\right)^{\mathsf{T}} \mathbf{1} = 0$, and $\boldsymbol{f}^{\mathsf{T}} D \boldsymbol{f} = \text{vol}\left(\mathcal{S}\right)$. A detailed deduction for this can be found in Appendix 2.A.2. As such, we can restate the normalised cut problem as a discrete minimisation problem involving the un-normalised graph Laplacian as follows

$$\begin{aligned}
\min_{\mathcal{S}_1,\mathcal{S}_2\in\mathcal{S}} \quad & \boldsymbol{f}^\mathsf{T} L \boldsymbol{f} \\
\text{s.t.} \quad & D\boldsymbol{f} \perp \mathbf{1}, \\
& \boldsymbol{f}^\mathsf{T} D \boldsymbol{f} = \text{vol}(\mathcal{S}), \\
& \boldsymbol{f} \text{ as defined in (2.3.12)}, \\
& \mathcal{S}_1 \cup \mathcal{S}_2 = \mathcal{S}, \quad \mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset.
\end{aligned} \tag{2.3.13}$$

Again, we consider a relaxation of the above problem in which the entries in $\boldsymbol{f}$ are allowed to take arbitrary values in $\mathbb{R}$. Through a change of variable $\boldsymbol{g} := D^{\frac{1}{2}}\boldsymbol{f}$, the relaxed problem can be restated using the symmetric normalised graph Laplacian matrix L_{sym} as

$$\begin{aligned}
\min_{\boldsymbol{g}\in\mathbb{R}^N} \quad & \boldsymbol{g}^\mathsf{T} L_{\text{sym}} \boldsymbol{g} \\
\text{s.t.} \quad & D^{\frac{1}{2}}\boldsymbol{g} \perp \mathbf{1}, \\
& \|\boldsymbol{g}\|_2^2 = \text{vol}\left(\mathcal{S}\right).
\end{aligned} \tag{2.3.14}$$

The solution to the above optimisation problem is given by the eigenvector of L_{sym} that corresponds to its second smallest eigenvalue. It is easy to check that λ is an eigenvalue of L_{rw} with eigenvector $\boldsymbol{v}$ if and only if λ is an eigenvalue of L_{sym} with eigenvector $D^{\frac{1}{2}}\boldsymbol{v}$. Therefore, $\boldsymbol{f}$ is the eigenvector of L_{rw} that corresponds to its second smallest eigenvalue. The discrete cluster labels can thus be found by applying K-means clustering to either $\boldsymbol{g}$ or $\boldsymbol{f}$.

For a general K-partitioning problem, the entries in the indicator matrix $H \in \mathbb{R}^{N\times K}$ for the normalised cut problem is specified as follows

$$H_{ik} = \begin{cases} \frac{1}{\sqrt{\text{vol}(\mathcal{S}_k)}}, & \boldsymbol{x}_i \in \mathcal{S}_k, \\ 0, & \text{otherwise}, \end{cases} \tag{2.3.15}$$

for $i \in \{1,\dots,N\}$ and $k \in \{1,\dots,K\}$. Following the same line of deduction as in the ratio cut setting, one can show that the following holds

$$H^\mathsf{T} H = I, \quad HDH = I, \quad \frac{\text{cut}\left(\mathcal{S}_k, \bar{\mathcal{S}}_k\right)}{\text{vol}\left(\mathcal{S}_k\right)} = \boldsymbol{h}_k^\mathsf{T} L \boldsymbol{h}_k,$$

for $k \in \{1, \ldots, K\}$. A detailed deduction for this can be found in Appendix 2.A.2. We now restate the normalised cut problem for general K as the following discrete trace minimisation problem

$$\begin{aligned} \min_{\mathcal{S}_1,\ldots,\mathcal{S}_K \subset \mathcal{S}} \quad & \operatorname{tr}(H^\mathsf{T} L H) \\ \text{s.t.} \quad & H^\mathsf{T} D H = I, \\ & H \text{ as defined in (2.3.15)}, \\ & \bigcup_{k=1}^{K} \mathcal{S}_k = \mathcal{S}, \\ & \mathcal{S}_i \cap \mathcal{S}_j = \emptyset, \quad \forall i,j \in \{1,\ldots,K\}. \end{aligned} \tag{2.3.16}$$

Relaxing the discreteness condition on H and apply the change of variable $T = D^{\frac{1}{2}} H$, we obtain the following relaxed problem involving the symmetric normalised graph Laplacian matrix L_{sym} as

$$\begin{aligned} \min_{T} \quad & \operatorname{tr}(T^\mathsf{T} L_{\text{sym}} T) \\ \text{s.t.} \quad & T^\mathsf{T} T = I. \end{aligned} \tag{2.3.17}$$

Again, the solution to this problem is given by the matrix $T^\star$ whose columns contain the K eigenvectors that correspond to the K smallest eigenvalues of L_{sym}. Similarly, H consists of the K eigenvectors of L_{rw} that correspond to its K smallest eigenvalues. The final cluster labels can be obtained by applying K-means clustering to the rows of H or T. The use of H corresponds to the symmetric spectral clustering algorithm (Ng et al., 2002), and the use of T corresponds to the random walk spectral clustering algorithm (Shi and Malik, 2000). A summary for the procedures of both normalised spectral clustering algorithms is provided in Algorithm 2 and 3.

A natural question that arises is: which of these two normalised graph Laplacians should we use? Furthermore, should we use the un-normalised graph Laplacian or the normalised graph Laplacians? To answer these questions, one can first check the degree distribution of the affinity matrix. If the degrees are evenly distributed, then there should not be a big difference in which graph Laplacian matrix is used. However if the opposite is true, then the normalised version is preferred over the un-normalised. This is because both of these two normalised graph Laplacians take into account the size of clusters and the within-cluster connectivity.

For both un-normalised and random walk spectral clustering, the eigenvectors of the corresponding graph Laplacian L and L_{rw} are used as the input to K-means clustering. Although this is also the case for symmetric spectral clustering, it is worth noting that λ is an eigenvalue of L_{rw} with eigenvector $\boldsymbol{v}$ if and only if λ is an eigenvalue of L_{sym} with eigenvector $D^{\frac{1}{2}}\boldsymbol{v}$. That is, the eigenvectors of L_{sym} are obtained by multiplying the eigenvectors of L_{rw} with $D^{\frac{1}{2}}$. This means that if some nodes have very small total edge weights, then the corresponding entries in the eigenvectors are very small as well. The arguments in Von Luxburg (2007) are in favour of the random walk spectral clustering for this reason. However, this issue of having small values in the eigenvectors are resolved by an additional row normalisation step in the symmetric spectral clustering algorithm. In addition, if a point has very weak connections to the remaining points in a data set, then there is reason to believe that it might be an outlier. Thus, the cluster label does not matter that much after all.

Algorithm 2: Spectral Clustering (Shi and Malik, 2000)

```
Input  : Data affinity matrix $A \in \mathbb{R}^{N \times N}$
         Number of clusters $K$
  1. Compute the normalised graph Laplacian: $L_{\text{rw}} = I - D^{-1}A$
  2. Compute the eigen-decomposition of $L_{\text{rw}}$
  3. Let $V \in \mathbb{R}^{N \times K}$ be the matrix whose columns contain the eigenvectors
     $\boldsymbol{v}_1, \dots, \boldsymbol{v}_K$ corresponding to the $K$ smallest eigenvalues of $L_{\text{rw}}$
  4. Group the rows of $V$ with the $K$-means algorithm into $K$ clusters
Output : Clusters $\{\mathcal{S}_1, \dots, \mathcal{S}_K\}$ with $\mathcal{S}_k = \{i | \omega_i = k\}$ for $k \in \{1, \dots, K\}$
```

2.3.4 Spectral Clustering - An Example

In this section, we apply spectral clustering to the two data sets with non-convex clusters that we used in Section 2.1 Figure 2.1.2. Previously, we have shown that K-means

```
Algorithm 3: Spectral Clustering (Ng et al., 2002)
  Input  : Data affinity matrix $A \in \mathbb{R}^{N \times N}$
           Number of clusters $K$
    1. Compute the normalised graph Laplacian: $L_{\text{sym}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$
    2. Compute the eigen-decomposition of $L_{\text{sym}}$
    3. Let $U \in \mathbb{R}^{N \times K}$ be the matrix whose columns contain the eigenvectors
       $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K$ corresponding to the $K$ smallest eigenvalues of $L_{\text{sym}}$
    4. Normalise the rows in $U$ to have unit length under the $\ell_2$-norm
    5. Group the normalised rows in $V$ with the $K$-means algorithm into $K$ clusters
  Output : Clusters $\{\mathcal{S}_1, \ldots, \mathcal{S}_K\}$ with $\mathcal{S}_k = \{i | \omega_i = k\}$ for $k \in \{1, \ldots, K\}$
```

clustering struggles to find a good partitioning of the data regardless of the chosen number of clusters. Here we show that spectral clustering is capable of finding the correct partitioning of the data on both examples.

As a first step, we need to determine the similarity graph thus construct the data affinity matrix. In Section 2.2.1, we discussed a few options for similarity graphs. On these two data sets, we experiment with both the k-NN graph and the fully connected graph with Gaussian similarity function. Both graphs have a tuning parameter: in k-NN graph, k controls the neighbourhood size; in Gaussian similarity function, σ controls the neighbourhood size. When the Gaussian similarity function is used as the proximity measure in the k-NN graph, we observe that both a k-NN graph with $k = 10$ and a fully connected graph with $\sigma = 100$ lead to the correct data partitioning. For illustration purpose, we will continue our discussion using the k-NN graph with $k = 10$.

Once the data affinity matrix is constructed, the next choice to make is which graph Laplacian matrix to construct. We discussed un-normalised spectral clustering, random walk spectral clustering, and symmetric spectral clustering in Section 2.3.2 and 2.3.3. We mentioned that the decision between un-normalised and normalised spectral clustering is dependent on the degree distribution. It does not make much of a difference when the degrees are evenly distributed, otherwise normalised distribution is often preferred over the un-normalised version (Von Luxburg, 2007). Histograms for the degree distribution of both data sets are shown in Figure 2.3.1. It is clear to see that the two histograms exhibit a bell shape, which suggests that the degrees are far from evenly distributed. Therefore, we narrow the options down to normalised graph Laplacians. Out of the two normalised graph Laplacians L_{rw} and L_{sym}, we choose to use L_{sym} as its eigenvectors account for the degree distribution.

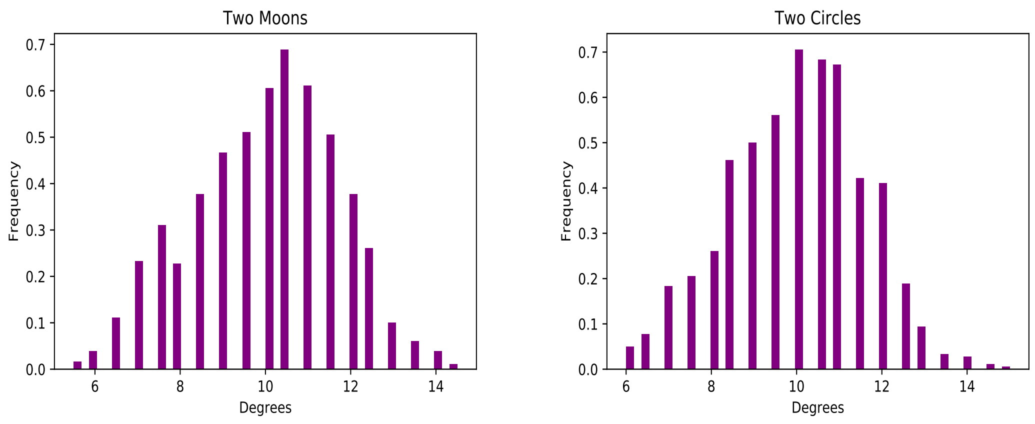


Figure 2.3.1: Histograms of the degree distribution based on the affinity matrix.

We apply eigen-decomposition to each of the two symmetric graph Laplacians, and obtain its $K = 2$ smallest eigenvalues and their corresponding eigenvectors. A visualisation of the first and second eigenvectors are provided in Figure 2.3.2. The points are coloured in their true cluster labels. It can be seen that the data are clearly separable in both data examples. We apply K-means clustering to the two eigenvectors after row normalisation, and a visualisation of the data coloured in the assigned cluster labels can also be found in the bottom row of Figure 2.3.2. We can see that spectral clustering has correctly identified the clusters on both data examples.

2.4 Subspace Clustering

So far, we have introduced K-means clustering and spectral clustering, which are two of the most fundamental approaches in clustering. However, the performance of these methods suffers in the presence of high-dimensionality, as is previously mentioned in Chapter 1. The curse-of-dimensionality, along with the exponential increase in the amount of high-dimensional data in recent years, largely motivated an active research

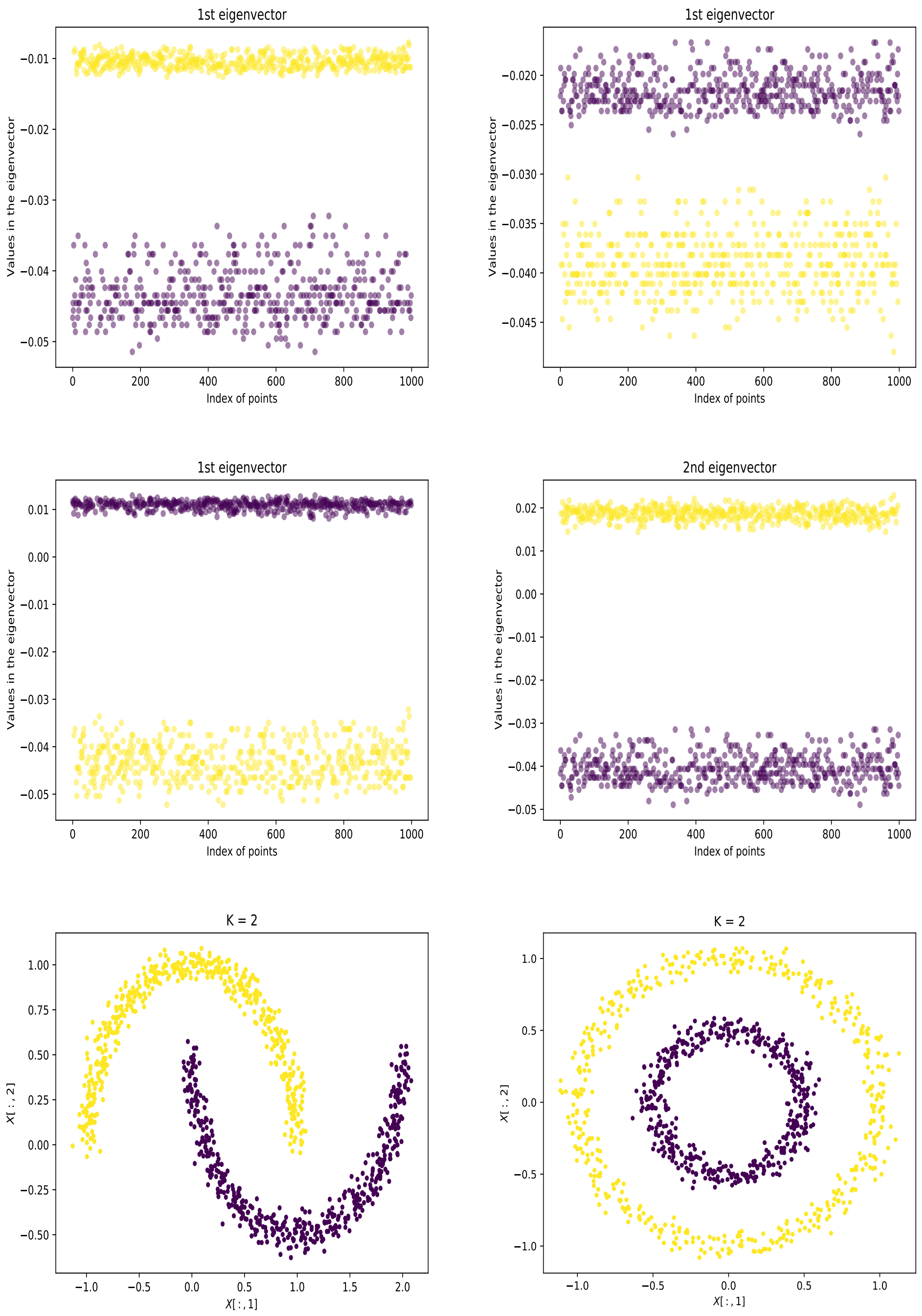


Figure 2.3.2: A visualisation of the two eigenvectors corresponding to the two smallest eigenvalues of L_{sym} (first and second row), and the data points coloured in the assigned cluster labels (third row).

area called *subspace clustering*. Subspace clustering is motivated by the observation that high-dimensional data can often be summarised well in a much lower-dimensional subspace (Elhamifar and Vidal, 2013).

Existing subspace clustering methods mainly fall under four different categories: iterative methods, spectral methods, algebraic methods, and statistical methods. We provide an overview of these different types of methods in the remainder of this section. In particular, our methodological contributions from Chapter 3 to 5 are based upon iterative, spectral, and algebraic methods respectively.

2.4.1 Iterative Methods

Iterative subspace clustering methods alternate between assigning points to subspaces and estimating the corresponding subspaces given the cluster labels (Bradley and Mangasarian, 2000; Tseng, 2000; Agarwal and Mustafa, 2004; Wang et al., 2009). Note that the general concept of iterative subspace methods resembles that of K-means clustering, except that the cluster centroid is replaced by the subspace basis.

K**-Plane Clustering (KPC)** (Bradley and Mangasarian, 2000) is a generalisation of K-means clustering from modelling the data as groups of spherical clusters to modelling the data as drawn from multiple hyperplanes,

$$\mathcal{P}_k = \left\{ \boldsymbol{x} \in \mathbb{R}^P | \boldsymbol{x}^\mathsf{T} \boldsymbol{v}_k = \gamma_k \right\}, \quad \forall\, k \in \{1, \dots, K\}. \tag{2.4.1}$$

The authors showed that the solution $(\boldsymbol{v}_k, \gamma_k)$ that gives the minimising hyperplane for a group of points is given by the smallest eigenvalue and its corresponding eigenvector of the following symmetric positive semi-definite matrix,

$$B_k = X_k^\mathsf{T} \left(I - \frac{1}{n_k} \mathbf{1}\mathbf{1}^\mathsf{T} \right) X_k, \tag{2.4.2}$$

where the rows of $X_k \in \mathbb{R}^{n_k \times P}$ correspond to the n_k points that are assigned to cluster k. In Eq. (2.4.1), $\boldsymbol{v}_k$ is given by the eigenvector that corresponds to the smallest eigenvalue of B_k as given in Eq. (2.4.2), and γ_k is given by $\gamma_k = \frac{1}{n_k}\mathbf{1}^\mathsf{T} X_k \boldsymbol{v}_k$. Similar to K-means clustering, the algorithm iterates between assigning each point to the closest hyperplane and estimating the hyperplane for each cluster of points.

K**-Subspace Clustering (KSC)** is one of the most popular iterative methods that has been invented and reinvented several times (Tseng, 2000; Agarwal and Mustafa, 2004; Wang et al., 2009). Tseng (2000) first generalised both K-means clustering which considers the cluster centre as a point, and K-plane clustering which represents each cluster as a hyperplane, to that of K-subspace clustering (KSC). Given that the data lie in a P-dimensional ambient space, KSC represents each cluster with a q-dimensional subspace ($0 \leqslant q \leqslant (P-1)$). It is easy to see that both K-means ($q=0$) and K-planes ($q = P-1$) are two special cases of this generic framework.

We present the base algorithm as described in both Agarwal and Mustafa (2004) and Wang et al. (2009) as follows. Given the set of N data points $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^N$, the aim is to find a set of subspace bases $\mathcal{V} = \left\{V_k \in \mathbb{R}^{P\times q}\right\}_{k=1}^K$ and cluster labels $\{\omega_i\}_{i=1}^N$ such that the overall reconstruction error is minimised. Concretely, the loss function can be expressed as follows

$$L(\mathcal{X};\mathit{\Omega}) = \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in \mathit{\Omega}_k} \|\boldsymbol{x}_i - V_{\omega_i} V_{\omega_i}^\mathsf{T} \boldsymbol{x}_i\|_2^2, \tag{2.4.3}$$

in which the number of subspaces K is assumed to be known. Given a set of cluster labels, the subspace bases $\mathcal{V} = \{V_1, \ldots, V_K\}$ can be obtained by applying Principal Component Analysis (PCA) (Jolliffe, 2011) to each group of data points from the same subspace such that the total reconstruction error in Eq. (2.4.3) is minimised. The columns in V_k are the top-q principal components for the k-th subspace. The basis matrix V_k for each subspace k can be obtained through the eigen-decomposition of its covariance matrix as

$$(X_k - \mathbf{1}\boldsymbol{\mu}_k^\mathsf{T})^\mathsf{T}(X_k - \mathbf{1}\boldsymbol{\mu}_k^\mathsf{T}) = V_k^\star \Lambda_k^\star (V_k^\star)^\mathsf{T}, \tag{2.4.4}$$

in which $X_k \in \mathbb{R}^{n_k \times P}$ is denoted as the data matrix that contains the n_k points assigned to cluster k, and $\boldsymbol{\mu}_k$ as the column-wise mean vector of X_k. $V_k^\star$ is a $P \times P$ matrix whose columns correspond to the eigenvectors of the covariance matrix of X_k, and $\Lambda_k^\star$ is a diagonal matrix containing the P eigenvalues.

Given the subspace bases $\mathcal{V} = \{V_1, \ldots, V_K\}$, the cluster label ω_i for a point $\boldsymbol{x}_i \in \mathcal{X}$

can be obtained as

$$\omega_i = \underset{k \in \{1,\ldots,K\}}{\arg\min} \left\| \boldsymbol{x}_i - V_k V_k^{\mathsf{T}} \boldsymbol{x}_i \right\|_2^2 . \tag{2.4.5}$$

KSC initialises with a set of randomly assigned cluster labels. The iterative process alternates between estimating the subspaces basis vectors according to Eq. (2.4.4), and updating the cluster labels according to Eq. (2.4.5). The algorithm terminates when the loss function value in Eq. (2.4.3) stops decreasing, and it is guaranteed to converge to a local optimum as is the case for the standard K-means clustering. It is worth noting that the subspace dimensions are assumed to be known and equal in the KSC base algorithm. However, this does not necessarily have to be the case. In Agarwal and Mustafa (2004), the authors extend KSC by introducing a dimension normalisation function for the determination of the corresponding subspace dimension q_k for each subspace k ($k \in \{1, \ldots, K\}$). This function captures the trade-off between the reconstruction error and the subspace dimension.

Median K-Flats (MKF) (Zhang et al., 2009) is an online iterative subspace clustering method that minimises the ℓ_1 distance between a point to the corresponding subspace as opposed to the ℓ_2 distance used in KSC. It has been observed that using the ℓ_1 distance measure is more robust than its ℓ_2 counterpart in the existence of a large number of outliers in the data. As in KSC, MKF also requires that all subspace dimensions are known and equal.

Instead of minimising the objective function in an iterative manner between subspace estimation and cluster assignment, MKF uses stochastic gradient descent to minimise the objective function (Christopher, 2006). The algorithm initialises with randomly allocated points and their corresponding K subspaces. At each iteration, a random point is chosen and allocated to its closest subspace, then the subspace is updated with stochastic gradient descent. The process repeats until some convergence criterion is met.

Although iterative methods are conceptually simple to implement and computationally fast to converge, they are sensitive to initialisation just like K-means clustering (Elhamifar and Vidal, 2013). An initialisation scheme called *farthest insertion* is proposed in Zhang et al. (2009), which has been shown to improve the cluster performance when the data have little noise and few outliers. Based on the same idea as farthest insertion, more recently He et al. (2016) proposed a robust algorithm for K-subspace recovery in the existence of outliers. The initialisation scheme first generates Q candidate subspaces ($Q \gg K$) using probabilistic farthest insertion (Ostrovsky et al., 2013). Then K out of these Q subspaces are selected based on a greedy algorithm.

Another approach to address the initialisation issue in iterative methods is to run the algorithm multiple times and then aggregate the results. Lipor et al. (2017) proposed **Ensemble K-Subspaces (EKSS)** which combines KSC with ensemble clustering. The main idea therein is to run KSC with multiple random initialisations, and combine the clustering results from multiple runs together. The development of EKSS is based on the observation that even those bad initialisations of KSC yield some partially correct cluster labels. As such, several KSC runs with bad initialisations may be combined to form a more accurate partitioning. Given a set of N data points, the EKSS algorithm performs KSC for a number of times with random initialisation, then forms an N by N co-association matrix in which the (i, j)-th entry of the matrix denotes the number of times point i and j are assigned to the same cluster. This co-association matrix is then modified by retaining only the top-q values from either each row or each column, in which q is a user-specified parameter. The purpose of this operation is to form a q nearest neighbours graph, so that ideally each point is connected to points from the same subspace (Heckel and Bölcskei, 2015). Spectral clustering is applied to the modified co-association matrix to obtain the final cluster labels.

Lipor (2017) provides theoretical guarantees on the performance of EKSS. Given EKSS with a specific choice of parameter values and by combining the clustering results from many random initialisations of KSC, it can be shown that the entries in the co-association matrix converge to a monotonically increasing function of the absolute value of the inner product between pairs of points (Lipor et al. (2017, Lemma 1)). In addition, it can cluster the points exactly under given conditions on the number of points per subspace and the maximum affinity between subspaces. Specifically, the affinity between subspace $\mathcal{S}_i$ and $\mathcal{S}_j$ is defined as (Heckel and Bölcskei, 2015; Zhang and Balzano, 2016)

$$\text{aff}\,(\mathcal{S}_i, \mathcal{S}_j) = \frac{1}{\min\,(q_i, q_j)} \|V_i^{\mathsf{T}} V_j\|_F, \tag{2.4.6}$$

where q_i and q_j denote the subspace dimensions for $\mathcal{S}_i$ and $\mathcal{S}_j$, V_i and V_j denote the subspace bases for $\mathcal{S}_i$ and $\mathcal{S}_j$. The EKSS algorithm has been shown to perform well on a number of benchmark data sets. Although it can be suitably parallelised due to its design, it does require more computing power to counteract the effect of bad initialisations.

2.4.2 Spectral Methods

In this section, we provide a brief review of spectral-based subspace clustering methods. A detailed discussion of state-of-the-art spectral-based methods will be provided in Chapter 4 Section 4.2.1. For an extensive overview of this area, we refer the reader to Vidal (2011). We will signpost more recent literature in the rest of this section.

Spectral-based methods are based upon the *self-expressiveness model* (Elhamifar and Vidal, 2013). Given a data set $\mathcal{X} = \{\boldsymbol{x}_i \in \mathbb{R}^P\}_{i=1}^N$, the main premise of this model is that every point $\boldsymbol{x}_i \in \mathcal{X}$ can be well approximated by a linear combination of a few other points from the same subspace. In the noise-free case, each point can be reconstructed by using exactly q points from the same linear subspace or $(q+1)$ points from the same affine subspace, in which q is the subspace dimension. Concretely, the self-expressiveness model can be expressed as the following optimisation problem for each point:

$$\begin{aligned} \min_{\boldsymbol{\beta}_i} \quad & \|\boldsymbol{\varepsilon}_i\|_\kappa + \rho\,\|\boldsymbol{\beta}_i\|_l \\ \text{s.t.} \quad & \boldsymbol{x}_i = Y_{-i}\boldsymbol{\beta}_i + \boldsymbol{\varepsilon}_i, \end{aligned} \tag{2.4.7}$$

where $Y_{-i} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{P\times(N-1)}$, i.e. the data matrix without the i-th column for $\boldsymbol{x}_i$ to prevent the trivial solution of self-representation, and ρ is a penalty parameter on the coefficient vector. Here $\boldsymbol{\beta}_i$ denotes the coefficient vector of the linear combination in representing $\boldsymbol{x}_i$, and $\boldsymbol{\varepsilon}_i$ represents the difference between $\boldsymbol{x}_i$ and the linear combination $Y_{-i}\boldsymbol{\beta}_i$, which is the *reconstruction error* term.

Combining the coefficient vectors for all N points together, we obtain the *coefficient matrix* B as follows

$$B = \begin{bmatrix} 0 & \beta_{12} & \beta_{13} & \dots & \beta_{1N} \\ \beta_{21} & 0 & \beta_{23} & \dots & \beta_{2N} \\ \beta_{31} & \beta_{32} & 0 & \dots & \beta_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta_{N1} & \beta_{N2} & \beta_{N3} & \dots & 0 \end{bmatrix}, \tag{2.4.8}$$

where β_{ij} in B denotes the coefficient value in front of $\boldsymbol{x}_i$ in the linear combination that approximates $\boldsymbol{x}_j$. That is, the coefficient vectors are stored in the columns of B. Given the coefficient matrix B, a common way to construct a non-negative symmetric affinity matrix A is through $A = \left(|B| + |B|^\intercal\right)/2$ (Huang et al., 2015; Li et al., 2017, 2018a). The final clustering labels can then be obtained by applying a standard spectral clustering algorithm (Shi and Malik, 2000; Ng et al., 2002) to the affinity matrix.

Most spectral-based methods differ in the choice of the norms $\|\cdot\|_\kappa$ and $\|\cdot\|_l$. The most influential work in the area of spectral-based subspace clustering is *Sparse Subspace Clustering (SSC)* (Elhamifar and Vidal, 2009). SSC applies the ℓ_1-norm to the coefficient vectors to encourage sparse solutions. In addition, it treats the existence of noise and sparse outlying entries with the ℓ_2 and ℓ_1-norm respectively. This is based on the fact that the data are often more evenly affected by noise, whereas sparse outlying entries are more local as is reflected in its name. Theoretical guarantee for the correctness of SSC is provided in Elhamifar and Vidal (2009), which shows that the solution vectors of SSC are subspace-preserving when the subspaces are independent (see Chapter 1 Section 1.2 for definition of independent subspaces). *Subspace-preserving* refers to the scenario where there is no connection between points from different subspaces, thus $\beta_{ij} \neq 0$ only when $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are in the same subspace (You et al., 2016; Li et al., 2018a). In Soltanolkotabi and Candes (2012), the correctness of SSC is further extended to the more general case where the subspaces could have non-trivial intersections (You, 2018).

Extensions of SSC. The success of SSC has led to the development of many other subspace clustering methods that also exploit the self-expressiveness property. Inspired by SSC, You et al. (2016) proposed a sparse subspace clustering method based on Orthogonal Matching Pursuit (OMP) (Pati et al., 1993). The proposed method is termed *SSC-OMP*, for its kinship to the original SSC algorithm. It uses OMP to recursively select one point at a time to minimise the ℓ_2-norm of the reconstruction error term, until a pre-specified k points are selected to be included into the sparse representation. That is, it applies the ℓ_0-norm on the coefficient vector β_i ($i \in \{1, \dots, N\}$). SSC-OMP is computationally efficient, thus suitable to be applied to large-scale problems. As in SSC, it has been shown that OMP gives a subspace-preserving representation of each data point if the subspaces are independent. In addition, SSC-OMP has also been shown to be subspace-preserving under certain conditions when the subspaces are not necessarily independent.

Another subspace clustering method that is based upon SSC is called *Structured Sparse Subspace Clustering (S3C)* (Li and Vidal, 2015; Li et al., 2017). A key difference between S3C and other spectral-based methods is that it integrates the stage of learning the representation matrix with the stage of spectral clustering. This results in an iterative optimisation framework. The optimisation programme in the first stage incorporates the results from spectral clustering through a new subspace structured ℓ_1-norm on the coefficient vector. This also paves the way for further incorporation of constraint information if there is any (Li et al., 2017, 2018b), which we will discuss in further detail in Chapter 4.

Dense representation models. Although certain conditions have been established for both SSC and SSC-OMP to be subspace-preserving, they are not sufficient conditions to produce correct clustering labels. Given that each point is represented by a few other points from the same subspace, it does not mean that all points in the same subspace form only one connected component. This may result in over-segmentation of points from one subspace by spectral clustering, which is known as the graph connectivity problem (Nasihatkon and Hartley, 2011; Wang et al., 2016).

To avoid this problem, one type of spectral-based method obtains the coefficient vector by minimising the ℓ_2-norm of the reconstruction error term in Eq. (2.4.7). For example, *Least Squares Regression (LSR)* (Lu et al., 2012) applies the ℓ_2-norm to both the reconstruction error term and the coefficient vector. The data affinity matrix obtained from the coefficient vectors are dense. Similar to LSR, *Smooth Representation Clustering (SMR)* (Hu et al., 2014) also minimises the least squares error on the reconstruction error term. In addition, it computes the graph Laplacian matrix L from a k-NN graph (See Section 2.2.1), and then incorporate L into the objective function. We provide more detailed discussion of the problem formulation in Chapter 4.

These ℓ_2-regularised problems have several nice properties. The objective of LSR corresponds to that of ridge regression, and has a closed form solution (Hoerl and Kennard, 1970). The objective of SMR is a smooth convex function, thus has a unique solution. Setting the derivative of the objective with respect to the solution vector yields the Sylvester equation, which can be solved by the Bartels-Stewart algorithm (Bartels and Stewart, 1972).

Another well-known dense representation model is *Low Rank Representation (LRR)* (Liu et al., 2010, 2012). It seeks a lowest rank representation of the data matrix with respect to a dictionary, which is the data matrix itself. It is mentioned in Liu et al. (2010) that the low rankness criterion is more suitable than the sparse representation one, as sparse representation models do not necessarily capture the global structure of the data. LRR solves a convex optimisation programme that applies the nuclear norm on the coefficient matrix, and the $\ell_{2,1}$-norm on the reconstruction error. The nuclear norm can be defined as the sum of all the singular values of a matrix. It is a convex envelop of the rank function, thus can serve as a convex surrogate for it. The $\ell_{2,1}$-norm is the sum of the ℓ_1-norms of the columns of a matrix. It encourages the columns of the reconstruction error term to be zero, which implies that the noise is specific to the data points.

Inclusion of an affine constraint. The aforementioned methods implicitly or explicitly deal with data that lie in linear subspaces. In general, data from affine subspaces can also be considered as in the case of linear subspaces. This is because a P-dimensional affine subspace $\mathcal{S}_{\text{aff}}$ can be considered as a $(P+1)$-dimensional linear subspace that includes $\mathcal{S}_{\text{aff}}$ and the origin (Elhamifar and Vidal, 2013). However, there are potential side effects of ignoring the affine structure of the data. In particular, it may result in indistinguishability between subspaces as a result of a potential increase in the dimension of the intersection between two subspaces.

To explicitly handle data from affine subspaces, one can include a constraint which requires that the coefficient vector sums up to one. This is based on the fact that any point from a q-dimensional affine subspace can be expressed with a combination of $(q + 1)$ other points from the same subspace (Elhamifar and Vidal (2013, Section 3.3)). The addition of an affine constraint to the SSC problem results in *Affine Sparse Subspace Clustering (ASSC)* (Li et al., 2018a). It has been observed that the ℓ_1-norm no longer induces sparsity with the inclusion of an affine constraint. As a result, the affinity matrix constructed from the coefficient matrix of ASSC is dense. It has been shown that ASSC enjoys subspace-preserving property when the affine subspaces are independent.

There are other methods that include an affine constraint in the problem formulation, but are not motivated by affine subspaces. *Sparse Simplex Representation (SSR)* is first proposed in Huang et al. (2013) for the modelling of brain networks. It includes the affine constraint as part of the simplex constraint to ensure that the resulting coefficient vectors are between zero and one, which provides a probabilistic interpretation for the coefficient values. In *Sparse Manifold Clustering and Embedding (SMCE)* (Elhamifar and Vidal, 2011), the aim is to identify points that lie in the same manifold. SMCE uses the geometrically motivated assumption that there exists a small neighbourhood for each point, in which only the points that come from the same manifold lie approximately in the same low-dimensional affine subspace. The proposed optimisation programme selects a few neighbours of each data point that span a low-dimensional affine subspace near that point. We will provide further details to these spectral-based subspace clustering methods in Chapter 4.

2.4.3 Algebraic Methods

Algebraic methods are mostly based on linear algebra, for example matrix factorisation; or polynomial algebra, which models the union of subspaces with a set of homogeneous polynomials. We discuss a few factorisation-based methods in Section 2.4.3.1 that are based on either the Singular Value Decomposition (SVD) or the Reduced Row Echelon Form (RREF) (Golub and Van Loan, 2013). In Section 2.4.3.2, we introduce the most influential work based on polynomial algebra – Generalised Principal Component Analysis (GPCA) (Vidal et al., 2003, 2005), and discuss some of the recent extensions based on GPCA.

2.4.3.1 Factorisation-based Methods

Methods that are based on matrix factorisation obtain the data segmentation from a low rank factorisation of the data matrix $X \in \mathbb{R}^{N \times P}$ (Vidal, 2011; Elhamifar and Vidal, 2013). Assume that the data matrix X_k which contains n_k points in the q_k-dimensional subspace k ($k \in \{1, \dots, K\}$) is noise-free, then we can express X_k in terms of its subspace basis vectors as

$$X_k = Y_k V_k^{\mathsf{T}}, \tag{2.4.9}$$

where $Y_k \in \mathbb{R}^{n_k \times q_k}$ is the low-dimensional representation of the data points in subspace k, and the columns of $V_k \in \mathbb{R}^{P \times q_k}$ correspond to the basis vectors for its q_k-dimensional subspace. If we order the rows of the full data matrix X according to their subspace labels as ΓX in which $\Gamma \in \mathbb{R}^{N \times N}$ is a permutation matrix, then we can express the ordered full data matrix ΓX as

$$\Gamma X = \begin{bmatrix} Y_1 & & & \\ & Y_2 & & \\ & & \ddots & \\ & & & Y_K \end{bmatrix} \cdot [V_1, \dots, V_K]^{\mathsf{T}} = YV. \tag{2.4.10}$$

If the subspaces are independent from each other, then we have $r := \text{rank}(X) = \sum_{k=1}^{K} q_k$, $Y \in \mathbb{R}^{N \times r}$, and $V \in \mathbb{R}^{r \times P}$. This idea of low rank matrix factorisation is the basis of a number of algebraic methods. For example, Boult and Brown (1991) and Costeira and Kanade (1998) rely on the Singular Value Decomposition (SVD) of X, and Gear (1998) utilises the Reduced Row Echelon Form (RREF) of X.

Most of these methods are motivated by the motion segmentation problem. The main objective of the motion segmentation problem is to identify a number of objects moving independently in three dimensions, captured by a sequence of two-dimensional images of a scene (Gear, 1998). Each rigid moving object can be described by a group of two-dimensional feature points $\{(x_1, y_1), (x_2, y_2), \ldots (x_{N_k}, y_{N_k})\}$. The pair (x_i, y_i) ($i \in \{1, \ldots, N_k\}$) contains the horizontal and vertical coordinates of the points in one image, where N_k denotes the total number of points for object k. The motion of each object is captured by a sequence of of frames that each contains a group of these feature points. These feature points corresponding to each object can be suitably modelled as a set of linearly independent subspaces.

SVD-based factorisation methods. Boult and Brown (1991) uses a rank-r SVD to approximate the data matrix, $X \approx \hat{U}\hat{\Sigma}\hat{V}^{\mathsf{T}}$, where $\hat{U} \in \mathbb{R}^{N \times r}$, $\hat{\Sigma} \in \mathbb{R}^{r \times r}$, and $\hat{V} \in \mathbb{R}^{P \times r}$. Given K linearly independent motions, they observe that the rank r is given by $3K$. This is because the location of each object captured by a moving camera can be characterised with three-dimensional location coordinates. Thus, they reside in a three-dimensional affine space, or four-dimensional linear space under homogeneous coordinates. A segmentation of the motions can be obtained by applying a clustering algorithm to the rows of $\hat{U}$. It is worth noting that this is essentially equivalent to applying clustering on the dimension reduced data as represented by the top-r principal component vectors.

The Costeira and Kanade algorithm (Costeira and Kanade, 1995, 1998) is also based upon the SVD to address the motion segmentation problem. Let $X \approx \hat{U}\hat{\Sigma}\hat{V}^{\mathsf{T}}$ be the rank-r SVD approximation of the data matrix, then the pairwise affinity information can be captured via $Q = \hat{U}\hat{U}^{\mathsf{T}} \in \mathbb{R}^{N \times N}$. In the noise-free scenario, $Q_{ij} \neq 0$ if point i and j are in the same subspace and zero otherwise. The cluster labels can be obtained either by applying spectral clustering to Q, or by sorting and thresholding the entries in Q and forming a block-diagonal structure (Vidal, 2011). A main drawback of the algorithm is that it is very sensitive to noise, and it is difficult to find a suitable threshold (Kanatani, 2001).

RREF-based factorisation methods. Another type of factorisation-based method relies on the Reduced Row Echelon Form (RREF). One of the earliest RREF-based methods is proposed in Gear (1994) and further developed in Gear (1998), and is motivated by the motion segmentation application as well. A motion sequence is composed of f image frames, and each image frame contains a number of two-dimensional points that are associated with K independently moving objects in a three-dimensional space. The aim is to identify which points belong to which moving objects throughout these f frames.

The locations of the points are recorded using homogeneous coordinates $W = [\boldsymbol{w}_1, \dots, \boldsymbol{w}_N]^\mathsf{T}$, where $\boldsymbol{w}_i = [x_i, y_i, z_i, 1]^\mathsf{T}$ is the location for point i ($i \in \{1, \dots, N\}$). Each frame is the result of a different transformation $T_j \in \mathbb{R}^{2\times 4}$ of the coordinates in W for $j \in \{1, \dots, f\}$. Let $T = [T_1, \dots, T_f]^\mathsf{T}$, then the data matrix can be obtained as $X = WT^\mathsf{T}$ which is of size N by P in which $P = 2f$. The pairs of columns in X correspond to the horizontal and vertical coordinates of the projection of all N points in the image frames. In the noise-free scenario, X would have rank no greater than $4K$. The algorithm first obtains the reduced row echelon form based on the transpose of the data matrix, $F = \text{rref}(X^\mathsf{T})$, with partial pivoting through the Gauss-Jordan elimination process. An outline of the process can be found in Gear (1994, 1998), and is given below in Algorithm 4.

At the end of the process, the reduced row echelon form F would have the following canonical form

$$F = \begin{bmatrix} 1 & 0 & \dots & 0 & F_{1,(r+1)} & \dots & F_{1,N} \\ 0 & 1 & \dots & 0 & F_{2,(r+1)} & \dots & F_{2,N} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 1 & F_{r,(r+1)} & \dots & F_{r,N} \end{bmatrix}, \qquad (2.4.11)$$

in which r is the rank of F. The first r columns of F is a r by r identity matrix, where each row / column is called a *pivot* row /column. The *rank tolerance* parameter t_r in Algorithm 4 controls the resulting rank of F. If it is too large, there would only be a few pivot columns. If it is too close to zero, there would be more pivot columns than the rank of the noise-free data.

Algorithm 4: Reduced Row Echelon Form (RREF) with Partial Pivoting

```
Input: Transpose of the data matrix: $Y = X^{\mathsf{T}}$; rank tolerance parameter: $t_r$
Initialisation: $i = 1, j = 1$
while $i \leqslant 2f$ and $j \leqslant N$ do
    Switch rows so that $\max_{r \in \{i,\dots,2f\}} Y_{r,j}$ is in row $i$, where
    $Y_{r,j}$ denotes the entry in the $r$-th row and the $j$-th column of $Y$
    if $Y_{i,j} > t_r$ then
        Divide the $i$-th row by $Y_{i,j}$ ($Y_{i,j}$ is the pivot)
        for $c \in \{1, \dots 2f\}$ and $c \neq i$ do
            $Y_{c\cdot} \leftarrow Y_{c\cdot} - Y_{c,j} \times Y_{i\cdot}$, where
            $Y_{c\cdot}$ denotes the $c$-th row of $Y$
        end
        $i \leftarrow i + 1$
    end
    $j \leftarrow j + 1$
end
```

The locations of the non-zero values in the reduced row echelon form provide the grouping information. Any two columns in F which have non-zero elements in the same row are considered to belong to the same moving object, thus in the same cluster. However, when the data matrix is not noise-free, the non-pivot columns $F_{\cdot,(r+1)}$ to $F_{\cdot,N}$ are often filled with non-zero entries only. As such, another user-specified parameter called the *grouping tolerance* parameter is introduced to set small entries in the non-pivot columns to zero. These two parameters combined have been shown to be able to tolerate a moderate amount of noise in the data (Gear, 1998). However, the success of the algorithm is based on the assumption that the subspaces are independent (see Chapter 1 Section 1.2 for the definition of independent subspaces).

2.4.3.2 Generalised Principal Component Analysis (GPCA)

The main idea. Generalised Principal Component Analysis (GPCA) (Vidal et al., 2003, 2005) is an algebraic-geometric method for clustering data lying in a union of linear subspaces. It models a union of K subspaces with a set of homogeneous polynomials with degree K. We illustrate the main idea behind this with a simple example shown in Figure 2.4.1, in which the union of two subspaces $\mathcal{S}_1 = \{\boldsymbol{x} | x_1 = x_2 = 0\}$ and

$\mathcal{S}_2 = \{x_3 = 0\}$ lie in a three-dimensional space, with normal vectors $\boldsymbol{b}_1, \boldsymbol{b}_2 \in \mathbb{R}^3$ respectively.

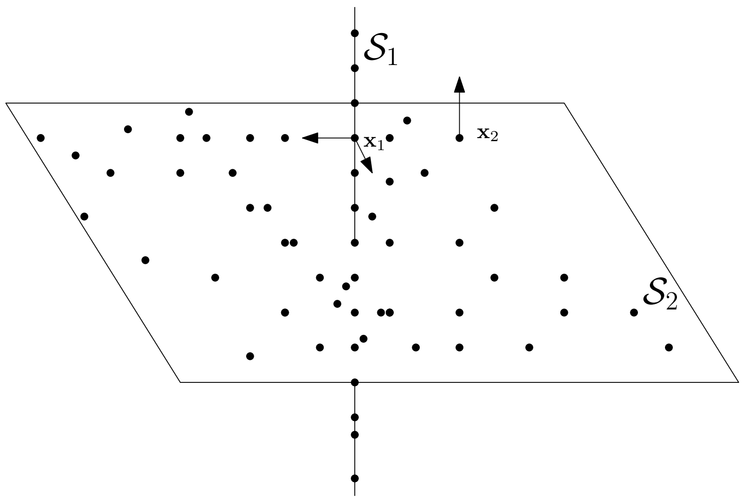


Figure 2.4.1: Data points drawn from a union of two subspaces in $\mathbb{R}^3$.

The union of the two subspaces can be characterised with the following polynomial

$$p(\boldsymbol{x}) = \left(\boldsymbol{b}_1^\mathsf{T}\boldsymbol{x}\right)\left(\boldsymbol{b}_2^\mathsf{T}\boldsymbol{x}\right) = 0, \tag{2.4.12}$$

where $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are the normal vectors that are orthogonal to points lying in subspaces $\mathcal{S}_1$ and $\mathcal{S}_2$ respectively. Eq. (2.4.12) says that the point $\boldsymbol{x} = [x_1, x_2, x_3]^\mathsf{T}$ either belongs to subspace $\mathcal{S}_1$ which gives $\boldsymbol{b}_1^\mathsf{T}\boldsymbol{x} = 0$, or belongs to subspace $\mathcal{S}_2$ which has $\boldsymbol{b}_2^\mathsf{T}\boldsymbol{x} = 0$.

As can be inspected visually, these two subspaces can also be characterised with two separate second-order polynomials

$$p_1(\boldsymbol{x}) = x_1 x_3 = 0, \quad p_2(\boldsymbol{x}) = x_2 x_3 = 0. \tag{2.4.13}$$

Let $P(\boldsymbol{x}) = [p_1(\boldsymbol{x}), p_2(\boldsymbol{x})]$ denote the set of two polynomials in Eq. (2.4.13), then the partial derivatives of $P(\boldsymbol{x})$ along all basis directions can be expressed as

$$\nabla P(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial P(\boldsymbol{x})}{\partial x_1} \\ \frac{\partial P(\boldsymbol{x})}{\partial x_2} \\ \frac{\partial P(\boldsymbol{x})}{\partial x_3} \end{bmatrix} = \begin{bmatrix} x_3 & 0 \\ 0 & x_3 \\ x_1 & x_2 \end{bmatrix}. \tag{2.4.14}$$

Consider two points $\boldsymbol{x}_1 = [0,0,1]^\mathsf{T} \in \mathcal{S}_1$ and $\boldsymbol{x}_2 = [1,1,0]^\mathsf{T} \in \mathcal{S}_2$ from these two subspaces respectively, the derivative of $P(\boldsymbol{x})$ at these two points can be written as

$$\nabla P(\boldsymbol{x}_1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \nabla P(\boldsymbol{x}_2) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}. \tag{2.4.15}$$

It can be seen the the columns of $\nabla P(\boldsymbol{x}_1)$ span the orthogonal complement of $\mathcal{S}_1$, which is denoted as $\mathcal{S}_1^\perp$. Similarly, the columns of $\nabla P(\boldsymbol{x}_2)$ span $\mathcal{S}_2^\perp$. It is also worth noting that the dimension of $\mathcal{S}_i$, plus the rank of its orthogonal complement given by the gradient of the set of polynomials, is equal to the ambient space dimension, i.e. $\text{rank}\,(\nabla P(\boldsymbol{x}_i)) + \dim\,(\mathcal{S}_i) = 3$ for $i \in \{1,2\}$. As such, if we can identify one point from each subspace, we can obtain the subspace bases and their dimensions from the gradient of $P(\boldsymbol{x})$ at these points.

Identifying one point per subspace can be conducted in a sequential manner. In the noise-free scenario, one can simply start by picking a random point from the data. When the data are noisy, one can pick a point that is closest to an existing subspace. The *algebraic distance* is used in Vidal et al. (2005), which is calculated as $d(\boldsymbol{x})^2 = p_1(\boldsymbol{x})^2 + p_2(\boldsymbol{x})^2 = (x_1^2 + x_2^2)\, x_3^2$ in our illustrative example. If we pick $\boldsymbol{x}_1 \in \mathcal{S}_1$ as the first point, then the next point can be obtained by dividing $\boldsymbol{b}_1^\mathsf{T}\boldsymbol{x}$ from the polynomial in Eq. (2.4.12). As such, we are left with $p(\boldsymbol{x}) = \boldsymbol{b}_2^\mathsf{T}\boldsymbol{x} = 0$, which can alternatively be expressed as the set of two polynomials $P(\boldsymbol{x}) = [x_1, x_2]$. The process of dividing one polynomial by another is called *polynomial division*.

The procedural form of GPCA. To summarise, the overall subspace clustering process of GPCA consists of three stages: polynomial fitting, differentiation, and division. As a first step, one needs to construct the polynomial with order K for a union of K subspaces. In order to do this, note that we can rewrite Eq. (2.4.12) in a general second-order polynomial as follows

$$c_1 x_1^2 + c_2 x_2^2 + c_3 x_3^2 + c_4 x_1 x_2 + c_5 x_1 x_3 + c_6 x_2 x_3 = 0. \tag{2.4.16}$$

Although this polynomial is non-linear in $\boldsymbol{x}$, it is linear in the coefficient vector $\boldsymbol{c} = [c_1, \ldots, c_6]^{\mathsf{T}}$.

In general the vector $\boldsymbol{c}$ is of length $M_K(P) = \binom{K+P-1}{K}$, which corresponds to the number of monomials in the order-K polynomial. Let $\nu(\boldsymbol{x}_i) \in \mathbb{R}^{M_K(P)}$ denote the vector of all monomials for $\boldsymbol{x}_i \in \mathcal{X}$, i.e. $\nu(\boldsymbol{x}_i) = [x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3]^{\mathsf{T}} \in \mathbb{R}^6$ for the second-order polynomial in Eq. (2.4.16). Then any point $\boldsymbol{x}_i$ from the union of subspaces satisfies the following homogeneous equation

$$\boldsymbol{c}^{\mathsf{T}}[\nu(\boldsymbol{x}_1), \ldots, \nu(\boldsymbol{x}_N)] = \boldsymbol{c}^{\mathsf{T}}V(\mathcal{X}) = \mathbf{0}^{\mathsf{T}}, \tag{2.4.17}$$

where $V(\mathcal{X}) \in \mathbb{R}^{M_K(P)\times N}$ is called the *embedded data matrix*. The space spanned by the set of all coefficient vectors that satisfy the homogeneous polynomial can be obtained from the SVD decomposition of the embedded data matrix.

Once this is done, the process proceeds by first picking a point that is closest to one of the subspaces and calculate the gradient of the polynomial at this point. Then we use polynomial division to characterise the union of the remaining $(K-1)$ subspaces. The process iterates between these two stages until all K subspaces are estimated and all points are assigned to their corresponding subspaces.

Discussion. GPCA addresses the subspace clustering problem in the most general case of an arbitrary number of subspaces with unknown and possibly different dimensions, and with arbitrary intersections among pairs of subspaces. It has been shown to perform favourably when compared to methods proposed in previous work. However, its computational complexity increases with the number of subspaces and their dimensions, and its performance deteriorates with the increase of the number of subspaces and their dimensions. The GPCA model is also very sensitive to the existence of noise and outliers. A robust extension to GPCA is proposed in Ma et al. (2008) which uses the Geometric Information criterion (Kanatani, 1998).

GPCA is originally designed only for data that lie in a union of linear subspaces. However, it can also be applied to data that are from affine subspaces by using homogeneous coordinates (Vidal, 2011). The homogeneous coordinates of $\boldsymbol{x} \in \mathbb{R}^P$ are given by

$\left[\boldsymbol{x}^\mathsf{T}, 1\right]^\mathsf{T} \in \mathbb{R}^{(P+1)}$. An affine subspace of dimension q in an ambient space of dimension P can be considered as a linear subspace of dimension $(q+1)$ in an ambient space of dimension $(P+1)$. Tsakiris and Vidal (2017) established the correctness of GPCA when applied to noise-free data lying in a union of affine subspaces.

2.4.4 Statistical Methods

Statistical methods make explicit assumptions about the distribution of the data and / or the distribution of the noise (Vidal, 2011). This type of method defines a generative model that is responsible for the observed data (Tipping and Bishop, 1999a; Gruber and Weiss, 2004; Yang et al., 2006; Archambeau et al., 2008; Rao et al., 2010; Arias-Castro et al., 2017).

2.4.4.1 Mixtures of Probabilistic Principal Component Analysers (MPPCA)

Principal component analysis (PCA) (Jolliffe, 2011) is one of the most widely used methods for dimension reduction and visualisation. However, it does not consider the data in a probabilistic framework, which makes it *ad hoc* to some extent. *Probabilistic Principal Component Analysis (PPCA)* (Tipping and Bishop, 1999b) puts PCA in a maximum-likelihood framework, which takes into account the probability density of the observed data. PPCA can be considered as a special case of statistical *factor analysis* (Bartholomew et al., 2011), which is one of the most popular latent variable models. Given a data set $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^N$, a latent variable model builds a relationship between each P-dimensional observed variable $\boldsymbol{x}$ and q-dimensional latent (unobserved) variable $\boldsymbol{y}$. Factor analysis assumes this relationship is linear, which can be expressed as

$$\boldsymbol{x} = V\boldsymbol{y} + \boldsymbol{\mu} + \boldsymbol{\varepsilon}, \tag{2.4.18}$$

in which the columns of $V \in \mathbb{R}^{P\times q}$ are the factor loadings, $\boldsymbol{\mu}$ is the feature-wise mean vector of the data, and $\boldsymbol{\varepsilon}$ is the noise in the data. The latent variable $\boldsymbol{y}$ is assumed to be Gaussian with zero mean and unit variance, $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{0}, I)$. The noise is also assumed to be

Gaussian distributed $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \Psi)$, in which the variance Ψ is a diagonal matrix. There is no closed form solution for obtaining V and Ψ, but an iterative Expectation-Maximisation (EM) algorithm can be used to estimate them (Tipping and Bishop, 1999b).

When $\Psi = \sigma^2 I$, i.e. all diagonal entries in Ψ are equal, PCA can be derived within the framework of density estimation. The specific noise distribution $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$ combined with Eq. (2.4.18) implies that the distribution of the observed variable $\boldsymbol{x}$ conditioned on the latent variable $\boldsymbol{y}$ is given by $\boldsymbol{x}|\boldsymbol{y} \sim \mathcal{N}(V\boldsymbol{y} + \boldsymbol{\mu}, \sigma^2 I)$. As such, we can obtain the PPCA model which has the following probability density function for any $\boldsymbol{x} \in \mathcal{X}$:

$$\begin{aligned} \mathbb{P}(\boldsymbol{x}) &= \int \mathbb{P}(\boldsymbol{x}|\boldsymbol{y})\mathbb{P}(\boldsymbol{y})d\boldsymbol{y} \\ &= (2\pi)^{\frac{P}{2}}|C|^{\frac{1}{2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathsf{T}}C^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right\}, \end{aligned} \tag{2.4.19}$$

where $C = VV^{\mathsf{T}} + \sigma^2 I$. A detailed derivation for Eq. (2.4.19) can be found in Tipping and Bishop (1999a) and Tipping and Bishop (1999b). Using Bayes' rule, we can obtain the conditional distribution of the latent variable $\boldsymbol{y}$ given the observed variable $\boldsymbol{x}$ as follows

$$\boldsymbol{y}|\boldsymbol{x} \sim \mathcal{N}\left(M^{-1}V^{\mathsf{T}}(\boldsymbol{x}-\boldsymbol{\mu}), \sigma^2 M^{-1}\right), \tag{2.4.20}$$

where $M = V^{\mathsf{T}}V + \sigma^2 I$. Note that $M \in \mathbb{R}^{q\times q}$, as compared to $C \in \mathbb{R}^{P\times P}$. It can be seen that when $\sigma \to 0$, the posterior mean $M^{-1}V^{\mathsf{T}}(\boldsymbol{x}-\boldsymbol{\mu})$ converges to $(V^{\mathsf{T}}V)^{-1}V^{\mathsf{T}}(\boldsymbol{x}-\boldsymbol{\mu})$. This represents an orthogonal projection onto the latent space, hence the conventional PCA is recovered.

In the PPCA model Eq. (2.4.19), V and σ^2 can be explicitly derived from maximum likelihood estimation as

$$V_{\text{ML}} = V_q\left(\Lambda_q - \sigma^2 I\right)^{\frac{1}{2}} R, \tag{2.4.21}$$

$$\sigma^2_{\text{ML}} = \frac{1}{P-q}\sum_{j=q+1}^{P} \lambda_j, \tag{2.4.22}$$

where Λ_q is the eigenvalue matrix whose diagonal entries contain the top-q eigenvalues

λ_js of the data, and V_q is the eigenvector matrix whose columns correspond to the top-q eigenvalues. Here R is an arbitrary $q \times q$ rotation matrix, which can be ignored for simplicity (i.e. $R = I$) (Tipping and Bishop, 1999b).

Mixtures of Probabilistic Principal Component Analysers (MPPCA) (Tipping and Bishop, 1999a) extends PPCA into a mixture of local PCA models, in which all of the model parameters may be estimated through the maximisation of a single likelihood function. The log-likelihood of observing the data set under the MPPCA model can be expressed as

$$\begin{aligned}\mathcal{L} &= \sum_{\boldsymbol{x} \in \mathcal{X}} \log \{\mathbb{P}(\boldsymbol{x})\} \\ &= \sum_{\boldsymbol{x} \in \mathcal{X}} \log \left\{ \sum_{k=1}^{K} \pi_k \mathbb{P}(\boldsymbol{x}|k) \right\},\end{aligned} \tag{2.4.23}$$

where $\mathbb{P}(\boldsymbol{x}|k)$ is a single PPCA model, and π_k is the proportion of data that are in the k-th sub-model (cluster), where $\sum_{k=1}^{K} \pi_k = 1$. The model parameters can be found via Expectation-Maximisation algorithm. The data points are first randomly allocated into K clusters. In the E-step, we calculate the probabilities of assigning each point $\boldsymbol{x}_i$ to all clusters k ($k \in \{1, \dots, K\}$), p_{ik}. Then, we assign each point to the cluster with the highest probability. In the M-step, p_{ik} is used to recompute the subspace parameters using PPCA. These two steps are iterated until convergence to a local maximum of the log-likelihood function.

The main advantage of MPPCA over iterative type of subspace clustering methods is that it provides a probabilistic framework for the data generation process. MPPCA can be considered as a probabilistic version of KSC, in which soft cluster assignments are used instead of hard cluster assignments. However, the advantage of MPPCA comes at the cost of a restrictive assumption that the distribution of the data and the noise have to be Gaussian, which is often not realistic in practice. Similar to KSC, MPPCA suffers from bad initialisations and is prone to converge to a local optimum.

In the same vein, Gruber and Weiss (2004) proposed a multi-body factorisation algorithm that is also formulated as a variant of factor analysis, which can be solved via the EM-algorithm. Unlike PPCA (and MPPCA), which has strict assumptions on the behaviour of the noise structure, the multi-body factorisation algorithm can handle arbitrary noise structure as well as missing data. More recently, Archambeau et al. (2008) proposed a robust version of PPCA and subsequently a mixture of robust PPCA models. It extends the MPPCA model to also take into account the existence of outliers by means of the Student's t-distribution to replace the Gaussian distribution.

2.4.4.2 Random Sample Consensus (RANSAC)

Another statistical method that addresses the issue of a significant amount of outliers is **Random Sample Consensus (RANSAC)** (Fischler and Bolles, 1981; Yang et al., 2006). RANSAC assumes that the data are drawn from a union of linear subspaces, and that the subspace dimensions must be known and equal (Yang et al., 2006). Unlike many other subspace methods that fit one model or a set of sub-models to the data as a whole, RANSAC fits one sub-model to a small number of points at a time.

RANSAC samples a small number of points at a time for enough times to reach a certain confidence level that one of these subsets is outlier-free or has very few outliers. This paradigm requires three parameters to be specified (Fischler and Bolles, 1981): (a) an error tolerance threshold beyond which a point is considered as an outlier given the model; (b) the total number of subsets of q points to sample from the whole data set, in which q is the known subspace dimension; and (c) the number / proportion of compatible points to suggest that the model is sound. The first subspace is estimated by repeatedly sampling q points from the data until (b) is violated or (c) is met. Otherwise, the sample with the largest number / proportion of compatible points is chosen. RANSAC proceeds in a greedy fashion by estimating one subspace at a time as follows:

(1) Estimate a new subspace, and assign a few points to the subspace. All remaining points in the data set are considered as outliers to the current subspace model.

(2) Remove the inliers of the previous subspace model from the current data set. Repeat step (1) to estimate the next subspace until all subspaces are estimated.

(3) Given the inliers in each subspace, use PCA to estimate the q basis vectors of the

subspace. Assign the points to the subspace that they have the smallest projection distance to.

The main advantage of RANSAC is that it addresses the presence of outliers explicitly. However, the performance of RANSAC deteriorates quickly with the increase in the number of subspaces. In addition, the computational complexity of the algorithm also increases exponentially with the dimension of the subspaces.

2.4.4.3 Agglomerative Lossy Compression (ALC)

Agglomerative Lossy Compression (ALC) is a simple clustering algorithm that models the data as a mixture of Gaussian distributions, which are allowed to be almost degenerate (Ma et al., 2007). Degeneracy of the data means that some features may be approximated by linear combinations of other features in the data. The original ALC algorithm is proposed in Ma et al. (2007), and it has been further extended in Rao et al. (2010) to solve the motion segmentation problem in the presence of outliers, missing entries, and corrupted trajectories.

ALC is an agglomerative clustering algorithm that proceeds in a bottom-up approach, hence the term "agglomerative" in its name. To begin with, each data point is treated as a group of its own. Then, two groups are merged that leads to the biggest decrease in the loss function. The algorithm terminates when the loss function cannot be decreased further through additionally merging any two existing groups.

The coding length function is used as the loss function, which provides a measure of goodness of the data segmentation. Coding length is the minimal number of bits needed to represent the data, subject to a given distortion of the data as determined by an allowable distortion parameter ε. Given a set of data points $\mathcal{X} = \left\{\boldsymbol{x}_i \in \mathbb{R}^P\right\}_{i=1}^N$, let $X_k \in \mathbb{R}^{P \times n_k}$ denote the data matrix whose columns correspond to the data vectors that are assigned to the k-th cluster. The loss (coding length) function for X_k can be expressed as follows

$$L\left(X_k, \varepsilon\right) = \frac{P + n_k}{2} \log_2 \left(\det\left(I + \frac{P}{n_k \varepsilon^2} X_k X_k^{\mathsf{T}}\right)\right), \tag{2.4.24}$$

which can be shown to be a smooth surrogate of rank(X_k) (Rao et al., 2010). Therefore, the objective in Eq. (2.4.24) can be considered as a surrogate for the rank minimisation problem. The overall loss function of ALC can be expressed as follows

$$L\left(\{X_k\}_{k=1}^{K}, \varepsilon\right) = \sum_{k=1}^{K} \left[L(X_k, \varepsilon) - n_k \log_2 \left(\frac{n_k}{N}\right)\right], \tag{2.4.25}$$

where the second term counts the number of bits needed to represent the labels of the data.

The algorithm only depends on a single parameter, the allowable *distortion* ε of the data. Once this is determined, the algorithm then automatically determines the number of the clusters, which does not involve any parameter estimation. The smaller ε is, the larger the number of clusters is, and vice versa. The optimal segmentation of the data should ideally result in the shortest coding length subject to a given distortion of the data.

Although ALC has been shown to work well in a number of motion segmentation examples, there is no systematic approach to choose the distortion parameter ε. In Rao et al. (2010), the authors propose to experiment with a range of ε values and pick the ones that produce the number of clusters that agree with our prior knowledge of the data set. This is in no way an efficient approach to determine the parameter value, as the computational cost of ALC is $\mathcal{O}(N^3 + N^2P^2 + NP^3)$. Furthermore, it is a greedy descent algorithm that does not guarantee a global convergence to the optimum of the loss function.

2.5 Clustering Performance Measures

In this section, we familiarise the reader with a few clustering performance measures that we will use throughout the remainder of this thesis. Performance measures evaluate how similar cluster assignment labels are to ground truth labels. It is often the case that the numbering of cluster assignment labels K do not match with that of the ground truth labels J. As long as all points that belong to the same cluster are assigned to the same label, and points that belong to different clusters are assigned to different labels, a perfect clustering is obtained.

However, when the clustering result is not perfect, there are several ways of measuring how close the cluster assignment labels are to the ground truth labels. The first two measures that we will introduce in this section are *purity* (Zhao and Karypis, 2001) and *clustering error* (Elhamifar and Vidal, 2013), which are two sides of the same coin. They evaluate the percentage of correctly and incorrectly clustered points respectively. In addition, we also introduce the *Adjusted Rand Index (ARI)* (Hubert and Arabie, 1985) and *Normalised Mutual Information (NMI)* (Cover and Thomas, 2012). These two measures are more advanced than the previous two, in that they also take into account the difference in the number of clusters, in addition to the conditional and joint entropy of both the ground truth labels and the cluster assignment labels.

2.5.1 Purity & Clustering Error

Purity (Zhao and Karypis, 2001) is one of the most straightforward measures to evaluate the performance of classification / clustering tasks. By counting the occurrences of the dominated label in each true class, it sums over the proportions of these occurrences across all J classes. Given a set of N points $\{\boldsymbol{x}_i\}_{i=1}^N$, let $\mathcal{C} = \{c_1, c_2, \ldots, c_N\}$ denote the set of ground truth labels for the data, and $\varOmega = \{\omega_1, \omega_2, \ldots, \omega_N\}$ the set of cluster assignment labels. Let us also denote $\mathcal{C}_j$ as the set of class labels correspond to class j ($j \in \{1, \ldots, J\}$), and $\varOmega_k$ the set of cluster labels that correspond to cluster k ($k \in \{1, \ldots, K\}$). Concretely, purity is calculated as follows

$$\text{Purity} = \frac{1}{N} \sum_{i=1}^{J} \max_j |\mathcal{C}_i \cap \varOmega_j|. \tag{2.5.1}$$

It gives us an intuitive understanding of how well a clustering algorithm performs given that the number of clusters K is known. With that said, it is not necessarily the case that the number of clusters K specified by a clustering algorithm agrees with the true number of classes J. It is worth noting that purity does not penalise for the number of clusters K. Consider the extreme case of assigning each data point to a cluster, this would produce a purity measure of 1. However, this in no way indicates that it is a perfect clustering result. It is simply an artefact of an extreme case of over-clustering. This would not happen for clustering methods that require the number of clusters to be known a priori.

As opposed to purity, clustering error measures the proportion of points that a clustering algorithm makes mistakes on. One can immediately obtain what the clustering error is once the purity score is known. It is calculated by

$$\text{Clustering error} = 1 - \frac{1}{N} \sum_{i=1}^{J} \max_{j} |\mathcal{C}_i \cap \Omega_j|. \tag{2.5.2}$$

Both of these two measures are commonly used to compare the algorithmic performance of different clustering algorithms.

2.5.2 Adjusted Rand Index (ARI)

Adjusted Rand Index (ARI) (Hubert and Arabie, 1985) is namely an adjusted measure of Rand Index (RI) (Rand, 1971). In order to introduce ARI, we need to first familiarise the reader with RI. Rand Index compares, for each pair of points, whether they belong to the same group or different groups according to the ground truth labels and according to the cluster assignment labels. Each pairwise comparison can be classified into one of the four scenarios as follows:

- a: the pair of points are classified into the same group both by the ground truth labels and by the cluster assignment labels,
- b: the pair of points are classified into the same group by the ground truth labels but into different groups by the cluster assignment labels,
- c: the pair of points are classified into the same group by the cluster assignment labels but into different groups by the ground truth labels,
- d: the pair of points are classified into different groups both by the ground truth labels and by the cluster assignment labels.

We denote the number of pairwise comparisons that fall into each of the four scenarios as n_a, n_b, n_c, and n_d respectively, then the Rand Index (RI) between the ground truth set $\mathcal{C}$ and the cluster assignment set Ω is calculated as

$$\mathrm{RI}\left(\mathcal{C},\Omega\right)=\frac{n_a+n_d}{n_a+n_b+n_c+n_d}. \tag{2.5.3}$$

A drawback of RI is that the expected value is not constant between two random partitions. To overcome this, Hubert and Arabie (1985) proposed the Adjusted Rand Index (ARI), which takes into account the expected value of the Rand index in the calculation. Table 2.1 summarises the pairwise comparisons between each true class $\mathcal{C}_j$ ($j \in \{1,\ldots,J\}$) and each assigned cluster Ω_k ($k \in \{1,\ldots,K\}$).

Classes / Clusters	$\lvert\mathcal{C}_1\rvert$	$\lvert\mathcal{C}_2\rvert$	$\ldots$	$\lvert\mathcal{C}_J\rvert$	$\sum_{j=1}^{J}\lvert\mathcal{C}_j\rvert$
$\lvert\Omega_1\rvert$	n_{11}	n_{12}	$\ldots$	n_{1J}	$n_{1\cdot}$
$\lvert\Omega_2\rvert$	n_{21}	n_{22}	$\ldots$	n_{2J}	$n_{2\cdot}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\lvert\Omega_K\rvert$	n_{k1}	n_{k2}	$\ldots$	n_{KJ}	$n_{K\cdot}$
$\sum_{k=1}^{K}\lvert\Omega_k\rvert$	$n_{\cdot 1}$	$n_{\cdot 2}$	$\ldots$	$n_{\cdot J}$	n

Table 2.1: Notation for comparing two set of labels on the same data set.

Using the count data in Table 2.1, the Adjusted Rand Index (ARI) between the ground truth set $\mathcal{C}$ and the cluster assignment set Ω is defined as

$$\mathrm{ARI}\left(\mathcal{C},\Omega\right)=\frac{\sum_{k,j}\binom{n_{kj}}{2}-\frac{\sum_k\binom{n_{k\cdot}}{2}\sum_j\binom{n_{\cdot j}}{2}}{\binom{n}{2}}}{\frac{\sum_k\binom{n_{k\cdot}}{2}+\sum_j\binom{n_{\cdot j}}{2}}{2}-\frac{\sum_k\binom{n_{k\cdot}}{2}\sum_j\binom{n_{\cdot j}}{2}}{\binom{n}{2}}}. \tag{2.5.4}$$

Note that the same term is subtracted from both the numerator and the denominator. This common term is the expected value of the Rand index. The first term in the denominator is the maximum index that can be obtained. Therefore, the expression for ARI can be considered as a corrected-for-chance version of the Rand index (Nguyen et al., 2009).

ARI takes a value between -1 and 1 that represents the amount of similarity in two clusterings. A value of 0 indicates the result is equivalent to random assignment, and a value close to 1 indicates strong agreement between the cluster assignment labels and the ground truth labels. A negative value means that the number of pairs of points that the cluster labels and the ground truth labels agree on is less than the expected number given by random assignment. Although ARI is an improvement on RI, it can be easily verified that two clustering results with the same purity score are likely to take on different ARI values. Hence it is best not to rely solely on ARI to compare the quality of two clustering results.

2.5.3 Normalised Mutual Information (NMI)

Normalised Mutual Information (NMI) (Cover and Thomas, 2012; Amelio and Pizzuti, 2015) is another commonly used measure to evaluate cluster performance. It is a normalised version of Mutual Information (MI), with higher NMI values indicating better clustering results. Following the notation in Table 2.1, the Mutual Information (MI) between the set of ground truth labels $\mathcal{C} = \{c_1, c_2, \ldots, c_N\}$ and the set of cluster assignment labels $\Omega = \{\omega_1, \omega_2, \ldots, \omega_N\}$ can be calculated as follows

$$\mathrm{MI}(\mathcal{C}, \Omega) = -\sum_{j=1}^{J}\sum_{k=1}^{K} \mathbb{P}(\mathcal{C}_j \cap \Omega_k) \log \frac{\mathbb{P}(\mathcal{C}_j \cap \Omega_k)}{\mathbb{P}(\mathcal{C}_j)\mathbb{P}(\Omega_k)}, \tag{2.5.5}$$

where $\mathbb{P}(\mathcal{C}_j)$, $\mathbb{P}(\Omega_k)$, and $\mathbb{P}(\mathcal{C}_j \cap \Omega_k)$ denote the proportions of points belonging to class j, cluster k, and both, respectively. This measure suffers from the same drawback as purity. That is, MI increases with the increase of the number of clusters K.

To resolve this problem, the improved NMI measure divides MI by the following normalising term:

$$\frac{H(\mathcal{C}) + H(\Omega)}{2}, \tag{2.5.6}$$

where $H(\Omega)$ is the entropy of the cluster assignment set Ω defined as follows,

$$H(\Omega) = -\sum_{k=1}^{K} \mathbb{P}(\Omega_k) \log \mathbb{P}(\Omega_k). \tag{2.5.7}$$

The entropy of the ground truth set $H(\mathcal{C})$ can be similarly obtained using the above formula. The normalising term averages the entropies of the ground truth labels and the cluster assignment labels. It penalises for the number of clusters, as the entropy is larger for larger number of clusters. Thus NMI is a measure that is always between 0 and 1, and we can also use it to compare clusterings with different number of clusters.

2.A Appendix: Connection between Graph Cuts and Graph Laplacians

2.A.1 The Ratio Cut and the Un-normalised Graph Laplacian

Given a general K-partitioning problem, the ratio cut objective is defined as follows

$$\text{RatioCut}(\mathcal{S}_1, \dots, \mathcal{S}_K) := \frac{1}{2}\sum_{k=1}^{K} \frac{W(\mathcal{S}_k, \bar{\mathcal{S}}_k)}{|\mathcal{S}_k|} = \sum_{k=1}^{K} \frac{\text{cut}(\mathcal{S}_k, \bar{\mathcal{S}}_k)}{|\mathcal{S}_k|}. \tag{2.A.1}$$

We can show that the ratio cut objective can be expressed as a discrete minimisation problem involving the graph Laplacian matrix.

For $K = 2$, the ratio cut objective of a bi-partitioning composed of two subsets $\{\mathcal{S}_1, \mathcal{S}_2\}$, in which $\mathcal{S}_1 \cup \mathcal{S}_2 = \mathcal{S}$ and $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$, can be expressed as

$$\begin{aligned}\text{RatioCut}(\mathcal{S}_1, \mathcal{S}_2) &= \sum_{k=1}^{2} \frac{\text{cut}(\mathcal{S}_k, \bar{\mathcal{S}}_k)}{|\mathcal{S}_k|} \\ &= \frac{\text{cut}(\mathcal{S}_1, \mathcal{S}_2)}{|\mathcal{S}_1|} + \frac{\text{cut}(\mathcal{S}_2, \mathcal{S}_1)}{|\mathcal{S}_2|} \\ &= \text{cut}(\mathcal{S}_1, \mathcal{S}_2) \cdot \left(\frac{1}{|\mathcal{S}_1|} + \frac{1}{|\mathcal{S}_2|}\right).\end{aligned} \tag{2.A.2}$$

Multiplying both sides of Eq. (2.A.2) by the cardinality of the set $|\mathcal{S}| = |\mathcal{S}_1| + |\mathcal{S}_2|$, we obtain

$$\begin{aligned}&|\mathcal{S}| \cdot \text{RatioCut}(\mathcal{S}_1, \mathcal{S}_2) \\ =&\text{cut}(\mathcal{S}_1, \mathcal{S}_2) \cdot \left(\frac{|\mathcal{S}_1| + |\mathcal{S}_2|}{|\mathcal{S}_1|} + \frac{|\mathcal{S}_1| + |\mathcal{S}_2|}{|\mathcal{S}_2|}\right) \\ =&\text{cut}(\mathcal{S}_1, \mathcal{S}_2) \cdot \left(\frac{|\mathcal{S}_2|}{|\mathcal{S}_1|} + \frac{|\mathcal{S}_1|}{|\mathcal{S}_2|} + 2\right) \\ =&\text{cut}(\mathcal{S}_1, \mathcal{S}_2) \cdot \left[\left(\sqrt{\frac{|\mathcal{S}_2|}{|\mathcal{S}_1|}}\right)^2 + \left(-\sqrt{\frac{|\mathcal{S}_1|}{|\mathcal{S}_2|}}\right)^2 - 2 \cdot \sqrt{\frac{|\mathcal{S}_2|}{|\mathcal{S}_1|}} \cdot \left(-\sqrt{\frac{|\mathcal{S}_1|}{|\mathcal{S}_2|}}\right)\right] \\ =&\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(f_i - f_j)^2 \\ =&\boldsymbol{f}^\mathsf{T} L \boldsymbol{f},\end{aligned}$$

in which the entries of $\boldsymbol{f}$ are defined as in (2.3.5). The last line from above follows by the definition of the un-normalised graph Laplacian,

$$\begin{aligned}
\boldsymbol{f}^\mathsf{T} L \boldsymbol{f} &= \boldsymbol{f}^\mathsf{T} D \boldsymbol{f} - \boldsymbol{f}^\mathsf{T} A \boldsymbol{f} \\
&= \sum_{i=1}^N f_i^2 d_i - \sum_{i=1}^N \sum_{j=1}^N f_i f_j w_{ij} \\
&= \frac{1}{2} \left(\sum_{i=1}^N d_i f_i^2 - 2 \sum_{i=1}^N \sum_{j=1}^N f_i f_j w_{ij} + \sum_{j=1}^N d_j f_j^2 \right) \\
&= \frac{1}{2} \left(\sum_{i=1}^N d_i f_i^2 - 2 \sum_{i=1}^N \sum_{j=1}^N f_i f_j w_{ij} + \sum_{j=1}^N d_j f_j^2 \right) \\
&= \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{ij} (f_i - f_j)^2.
\end{aligned}$$

It is easy to verify that $\boldsymbol{f}$ also satisfies the following conditions:

$$\begin{aligned}
&\sum_{i=1}^N f_i = \sum_{i \in \mathcal{S}_1} \sqrt{\frac{|\mathcal{S}_2|}{|\mathcal{S}_1|}} - \sum_{i \in \mathcal{S}_2} \sqrt{\frac{|\mathcal{S}_1|}{|\mathcal{S}_2|}} = |\mathcal{S}_1| \sqrt{\frac{|\mathcal{S}_2|}{|\mathcal{S}_1|}} - |\mathcal{S}_2| \sqrt{\frac{|\mathcal{S}_1|}{|\mathcal{S}_2|}} = 0, \\
&\|\boldsymbol{f}\|_2^2 = \sum_{i=1}^N f_i^2 = |\mathcal{S}_1| \frac{|\mathcal{S}_2|}{|\mathcal{S}_1|} + |\mathcal{S}_2| \frac{|\mathcal{S}_1|}{|\mathcal{S}_2|} = N.
\end{aligned} \tag{2.A.3}$$

Therefore, we can re-express the ratio cut objective for $K = 2$ as the following discrete optimisation problem:

$$\begin{aligned}
\min_{\mathcal{S}_1, \mathcal{S}_2 \in \mathcal{S}} \quad & \boldsymbol{f}^\mathsf{T} L \boldsymbol{f} \\
\text{s.t.} \quad & \boldsymbol{f} \perp \mathbf{1}, \\
& \|\boldsymbol{f}\|_2 = \sqrt{N}, \\
& \boldsymbol{f} \text{ as defined in (2.3.5)}, \\
& \mathcal{S}_1 \cup \mathcal{S}_2 = \mathcal{S}, \quad \mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset.
\end{aligned} \tag{2.A.4}$$

For general K-partitioning problems, we can express the ratio cut between $\mathcal{S}_k$ and its complement $\bar{\mathcal{S}}_k$ ($k \in \{1, \ldots, K\}$) as

$$\begin{aligned}
\frac{\text{cut}(\mathcal{S}_k,\bar{\mathcal{S}}_k)}{|\mathcal{S}_k|} &= \frac{1}{\sqrt{|\mathcal{S}_k|}}\text{cut}(\mathcal{S}_k,\bar{\mathcal{S}}_k)\frac{1}{\sqrt{|\mathcal{S}_k|}} \\
&= \frac{1}{2}\sum_{i\in\mathcal{S}_i, j\in\bar{\mathcal{S}}_i}\frac{w_{ij}}{\sqrt{|\mathcal{S}_k|}\sqrt{|\mathcal{S}_k|}} + \frac{1}{2}\sum_{i\in\bar{\mathcal{S}}_i, j\in\mathcal{S}_i}\frac{w_{ij}}{\sqrt{|\mathcal{S}_k|}\sqrt{|\mathcal{S}_k|}} \\
&= \frac{1}{2}\sum_{i,j=1}^{N}\boldsymbol{h}_k^{\mathsf{T}}\boldsymbol{h}_k w_{ij} \\
&= \boldsymbol{h}_k^{\mathsf{T}} L \boldsymbol{h}_k .
\end{aligned}$$

By aggregating the above term for all $\mathcal{S}_i$ we obtain

$$\text{RatioCut}(\mathcal{S}_1,\ldots,\mathcal{S}_K) = \sum_{k=1}^{K}\boldsymbol{h}_k^{\mathsf{T}} L\boldsymbol{h}_k = \sum_{k=1}^{K}(H^{\mathsf{T}}LH)_{kk} = \text{tr}(H^{\mathsf{T}}LH). \tag{2.A.5}$$

It is easy to verify that the columns in H are orthonormal to each other. Therefore, we can express the ratio cut objective for general K as the following discrete trace minimisation problem:

$$\begin{aligned}
\min_{\mathcal{S}_1,\ldots,\mathcal{S}_K\subset\mathcal{S}} \quad & \text{tr}(H^{\mathsf{T}}LH) \\
\text{s.t.} \quad & H^{\mathsf{T}}H = I, \\
& H \text{ as defined in (2.3.7)}, \\
& \bigcup_{k=1}^{K}\mathcal{S}_k = \mathcal{S}, \\
& \mathcal{S}_i\cap\mathcal{S}_j = \emptyset, \quad \forall i,j\in\{1,\ldots,K\}.
\end{aligned} \tag{2.A.6}$$

2.A.2 The Normalised Cut and the Normalised Graph Laplacians

Given a general K-partitioning problem, the normalised cut objective is defined as follows

$$\text{NCut}(\mathcal{S}_1,\ldots,\mathcal{S}_K) := \frac{1}{2}\sum_{k=1}^{K}\frac{W(\mathcal{S}_k,\bar{\mathcal{S}}_k)}{\text{vol}(\mathcal{S}_k)} = \sum_{k=1}^{K}\frac{\text{cut}(\mathcal{S}_k,\bar{\mathcal{S}}_k)}{\text{vol}(\mathcal{S}_k)}.$$

We can show that the normalised cut objective can be expressed as a discrete minimisation problem involving the graph Laplacian matrix.

For $K = 2$, using similar algebraic substitutions as in the ratio cut setting, we have

$$\begin{aligned}
\mathrm{vol}(\mathcal{S}) \cdot \mathrm{NCut}(\mathcal{S}_1, \mathcal{S}_2) &= \mathrm{vol}(\mathcal{S}) \sum_{k=1}^{2} \frac{\mathrm{cut}(\mathcal{S}_k, \bar{\mathcal{S}}_k)}{\mathrm{vol}(\mathcal{S}_k)} \\
&= \mathrm{cut}(\mathcal{S}_1, \mathcal{S}_2) \left[\frac{\mathrm{vol}(\mathcal{S}_2)}{\mathrm{vol}(\mathcal{S}_1)} + \frac{\mathrm{vol}(\mathcal{S}_1)}{\mathrm{vol}(\mathcal{S}_2)} + 2 \right] \\
&= \sum_{i \in \mathcal{S}_1, j \in \mathcal{S}_2} w_{ij} \left(f_i - f_j\right)^2 \\
&= \boldsymbol{f}^{\mathsf{T}} L \boldsymbol{f}.
\end{aligned} \tag{2.A.7}$$

It is easy to verify that $\boldsymbol{f}$ also satisfies the following conditions:

(1) $(D\boldsymbol{f})^{\mathsf{T}} \mathbf{1} = 0$,

(2) $\boldsymbol{f}^{\mathsf{T}} D \boldsymbol{f} = \mathrm{vol}(\mathcal{S})$.

Condition (1) can be shown through the following deduction:

$$\begin{aligned}
(D\boldsymbol{f})^{\mathsf{T}} \mathbf{1} &= \sum_{i=1}^{N} d_i f_i \\
&= \sum_{m \in \mathcal{S}_1} d_m f_m + \sum_{n \in \mathcal{S}_2} d_n f_n \\
&= \mathrm{vol}\left(\mathcal{S}_1\right) \sqrt{\frac{\mathrm{vol}\left(\mathcal{S}_2\right)}{\mathrm{vol}\left(\mathcal{S}_1\right)}} - \mathrm{vol}\left(\mathcal{S}_2\right) \sqrt{\frac{\mathrm{vol}\left(\mathcal{S}_1\right)}{\mathrm{vol}\left(\mathcal{S}_2\right)}} \\
&= 0.
\end{aligned}$$

Condition (2) can be shown through the following deduction:

$$\begin{aligned}
\boldsymbol{f}^{\mathsf{T}} D \boldsymbol{f} &= \sum_{i=1}^{N} d_i f_i^2 \\
&= \sum_{m \in \mathcal{S}_1} d_m f_m^2 + \sum_{n \in \mathcal{S}_2} d_n f_n^2 \\
&= \mathrm{vol}\left(\mathcal{S}_1\right) \frac{\mathrm{vol}\left(\mathcal{S}_2\right)}{\mathrm{vol}\left(\mathcal{S}_1\right)} + \mathrm{vol}\left(\mathcal{S}_2\right) \frac{\mathrm{vol}\left(\mathcal{S}_1\right)}{\mathrm{vol}\left(\mathcal{S}_2\right)} \\
&= \mathrm{vol}\left(\mathcal{S}\right).
\end{aligned}$$

Therefore, we can re-express the normalised cut objective for $K = 2$ as the following discrete optimisation problem:

$$\begin{aligned}
\min_{\mathcal{S}_1,\mathcal{S}_2\in\mathcal{S}} \quad & \boldsymbol{f}^\mathsf{T} L \boldsymbol{f} \\
\text{s.t.} \quad & D\boldsymbol{f} \perp \mathbf{1}, \\
& \boldsymbol{f}^\mathsf{T} D \boldsymbol{f} = \text{vol}(\mathcal{S}), \\
& \boldsymbol{f} \text{ as defined in (2.3.12)}, \\
& \mathcal{S}_1 \cup \mathcal{S}_2 = \mathcal{S}, \quad \mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset.
\end{aligned} \tag{2.A.8}$$

For general K-partitioning problems, in the same vein to the deduction for the ratio cut setting, we can express the normalised cut between $\mathcal{S}_k$ and its complement $\bar{\mathcal{S}}_k$ ($k \in \{1, \ldots, K\}$) as

$$\frac{\text{cut}\left(\mathcal{S}_k, \bar{\mathcal{S}}_k\right)}{\text{vol}\left(\mathcal{S}_k\right)} = \boldsymbol{h}_k^\mathsf{T} L \boldsymbol{h}_k. \tag{2.A.9}$$

Thus the following equivalence can be built for the normalised cut objective:

$$\text{NCut}(\mathcal{S}_1, \ldots, \mathcal{S}_K) = \sum_{k=1}^{K} \boldsymbol{h}_k^\mathsf{T} L \boldsymbol{h}_k = \sum_{k=1}^{K} (H^\mathsf{T} L H)_{kk} = \text{tr}(H^\mathsf{T} L H). \tag{2.A.10}$$

The columns in H are orthonormal to each other, and we have the following

$$\begin{aligned}
\boldsymbol{h}_k^\mathsf{T} D \boldsymbol{h}_k &= \sum_{i=1}^{N} h_{ik}^2 D_{ii} \\
&= \sum_{m\in\mathcal{S}_k} h_{mk}^2 d_m + \sum_{n\notin\mathcal{S}_k} h_{nk}^2 d_n \\
&= \sum_{m\in\mathcal{S}_k} \frac{1}{\text{vol}(\mathcal{S}_k)} d_m + \sum_{n\notin\mathcal{S}_k} 0 \times d_n \\
&= \frac{\text{vol}\left(\mathcal{S}_k\right)}{\text{vol}\left(\mathcal{S}_k\right)} \\
&= 1,
\end{aligned}$$

for $k \in \{1, \ldots, K\}$. Thus we have $HDH = I$.

Chapter 3

Subspace Clustering with Active Learning

Subspace clustering is a growing field of unsupervised learning that has gained much popularity in the computer vision community. Applications can be found in areas such as motion segmentation and face clustering. It assumes that the data points originate from a union of subspaces, and clusters the data depending on the corresponding subspace. In practice, it is reasonable to assume that a limited number of labels can be obtained, potentially at a cost. Therefore, algorithms that can effectively and efficiently incorporate this information to improve the clustering model are desirable. In this work, we propose an active learning framework for subspace clustering that sequentially queries informative points and updates the subspace model. The query stage of the proposed framework relies on results from the perturbation theory of Principal Component Analysis (PCA) to identify those influential and potentially misclassified points. A constrained subspace clustering algorithm is proposed that monotonically decreases the objective function subject to the constraints imposed by the labelled data. We show that our proposed framework is suitable for subspace clustering algorithms, including iterative methods and spectral methods. Experiments on synthetic data sets, motion segmentation data sets, and Yale Faces data sets demonstrate the advantage of our proposed active strategy over state-of-the-art methods.

3.1 Introduction

In recent years, crowdsourcing (Su et al., 2012) for data annotation has drawn much attention in the computer vision community, due to the need to make use of as much data as possible and the lack of sufficiently labelled data. Clustering is commonly used as an initial step to provide a coarse preliminary grouping in the absence of labelled data. For example, there are plant recognition apps that allow one to take a photo of a plant and identify its species. In video surveillance, one may wish to identify the points corresponding to an object that exists in a sequence of frames, be it people or cars etc. Usually some form of external information is available in these applications, either through crowdsourcing websites, or through paid manual work to conduct a limited amount of labelling. In either case, obtaining labels involves a cost which is either time, money, or both. Therefore, effective and efficient ways of carrying out data annotation are desirable.

The process of iteratively annotating the potentially misclassified data and subsequently updating the model is generally known as active learning (Settles, 2008). It is a subfield of machine learning that aims to improve both supervised and unsupervised algorithms. In supervised learning, points that are near the decision boundary are likely to be misclassified. In unsupervised learning, the notion of potentially misclassified points is less clear and is open for interpretation.

In subspace clustering, points are clustered according to their underlying subspaces. There are different ways of measuring how likely a point is misclassified. One approach is to consider points whose projection onto the associated subspace is large as potentially misclassified (Lipor and Balzano, 2015). Alternatively, points that are almost equidistant to their two nearest subspaces are likely to be misclassified (Lipor and Balzano, 2017). Such ideas are based on the notion of the *reconstruction error* between the original point and its projection to the subspace that defines the cluster. The total reconstruction error is the objective function of the K-Subspace Clustering (KSC) algorithm (Agarwal and Mustafa, 2004). We therefore argue that effective active learning strategies should explicitly associate the query procedure with the optimisation of this objective. However, as we will discuss in greater detail in the next section, the point with the largest reconstruction error is not necessarily the most informative from the perspective of updating the entire subspace clustering model.

Motivated by the connection between the reconstruction error and the KSC objective, we consider a point to be influential if querying its true class thus updating the cluster assignment can lead to a large decrease in the total reconstruction error. Given a set of cluster labels, the optimal linear subspace for each cluster can be trivially estimated through Principal Component Analysis (PCA) (Jolliffe, 2011). In particular, the basis for each subspace (cluster) can be defined through the set of eigenvectors of the covariance matrix of the points that are assigned to this cluster. We make use of ideas from the perturbation analysis of PCA (Critchley, 1985) to evaluate efficiently how influential each point is, and query the class of the most informative point(s). Once the true classes of the influential points have been identified, our proposed K-subspace clustering with constraints (KSCC) algorithm monotonically reduces the reconstruction error whilst satisfying all the constraints imposed by the labelled data. The active learning process iterates between these two query and update procedures until the query budget is exhausted.

The rest of this chapter is organised as follows. We review related work in active learning in Section 3.2, and introduce our proposed active framework in Section 3.3. Experimental results on synthetic and real data are presented and discussed in Section 3.4. The chapter finishes in Section 3.6 with conclusions and directions for future work.

3.2 Related Work

There are three main approaches to active learning (Settles, 2008): uncertainty sampling (Balcan et al., 2007), query by committee (Seung et al., 1992), and expected model change (Settles et al., 2008).

Uncertainty sampling queries the points the learning algorithm is least confident about. Classic uncertainty sampling methods are generally ignorant to the data distribution, thus prone to select outliers (Donmez et al., 2007). It is suggested in Melville and Mooney (2004) to measure the informativeness of each point by the probability margin between the label it is assigned to and its second most likely label. Other versions of uncertainty sampling have been proposed to balance the density of a region and the uncertainty in that region (Nguyen and Smeulders, 2004). When building supervised models, one may also choose the unlabelled points near the decision boundary.

Query by committee (QBC) is a type of active learning strategy designed for classifier ensembles (Seung et al., 1992). It constructs a committee of models based on the labelled training data, and chooses to query the unlabelled points upon which the predictions of the classifiers in the ensemble disagree the most. It enables the training of accurate classifiers using a small subset of the data. To use this strategy, one has to provide both the type of classifier and a measure of disagreement among the classifiers. It has been shown in Freund et al. (1997) that rapid decrease in the misclassification error is guaranteed if the queries have high expected information gain.

Expected model change is an active learning framework that bases its query strategy on the idea that a point is informative if knowing its true class can cause a big change in the current model (Settles et al., 2008). This is mostly applied to discriminative probabilistic modelling, in which the gradient of the model is used as an indicator for the informativeness of a point. It is widely applied to image retrieval and text classification (Roy and McCallum, 2001). The method we propose also adopts this approach, but to the best of our knowledge, we are the first to consider updating an unsupervised learning model describing all the data, rather than just the labelled data.

As is implied above, most active learning approaches have been developed for supervised learning. However, less attention has been paid to the unsupervised counterpart. Only a few active learning strategies have been proposed for subspace clustering (Lipor and Balzano, 2015, 2017). In Lipor and Balzano (2015), two active strategies *MaxResid* and *MinMargin* for KSC are proposed. *MaxResid* queries points that have large reconstruction error to their allocated subspaces. *MinMargin* queries points that are maximally equidistant to their two closest subspaces. These two strategies are effective in identifying the points that are most likely to be misclassified. However, the queried points are not necessarily the most informative points in terms of updating the full clustering model.

3.3 Active Learning Framework

In this section, we first formulate the subspace clustering problem and then present the proposed active learning framework. There are two iterative procedures within this framework. The first is to identify the most influential and potentially misclassified points. The second is to update the cluster labels for all data points given the labelling information.

3.3.1 K-Subspace Clustering

A q-dimensional linear subspace $\mathcal{S}_k$, $k \in \{1, \dots, K\}$, can be defined through an orthonormal matrix $V_k \in \mathbb{R}^{P \times q}$ as

$$\mathcal{S}_k = \left\{ \boldsymbol{x} \in \mathbb{R}^P : \ \boldsymbol{x} = V_k \boldsymbol{y} \right\}, \tag{3.3.1}$$

where the columns of V_k constitute a basis for $\mathcal{S}_k$.

In subspace clustering, the overall objective is to find the set of optimal cluster labels for all data points such that the total reconstruction error between each data point to their corresponding subspaces is minimised. Given the set of N data points $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^N$ and their cluster labels $\Omega = \{\omega_i\}_{i=1}^N$, the loss function value $L(\boldsymbol{x}_i, V_{\omega_i})$ and the objective $f(\mathcal{X}, \mathcal{V})$ can be written as

$$L(\boldsymbol{x}_i, V_{\omega_i}) = \|\boldsymbol{x}_i - V_{\omega_i} V_{\omega_i}^\top \boldsymbol{x}_i\|_2^2, \tag{3.3.2}$$

and

$$f(\mathcal{X}, \mathcal{V}) = \min_{V_1, \dots, V_K} \sum_{i=1}^N \min_{\omega_i \in \{1, \dots, K\}} L(\boldsymbol{x}_i, V_{\omega_i}), \tag{3.3.3}$$

where $\mathcal{V} = \{V_1, \dots, V_K\}$ represents the set of all subspace bases. This objective can be minimised through a K-means-like iterative algorithm by alternating between *subspace estimation* and *cluster assignment*.

We need to obtain the set of subspace bases such that the total reconstruction error in Eq. (3.3.3) is minimised. The basis matrix V_k for each subspace k can be obtained through the eigen-decomposition of its covariance matrix as

$$(X_k - \mathbf{1}\boldsymbol{\mu}_k^\mathsf{T})^\mathsf{T}(X_k - \mathbf{1}\boldsymbol{\mu}_k^\mathsf{T}) = V_k^\star \Lambda_k^\star (V_k^\star)^\mathsf{T}. \tag{3.3.4}$$

We denote $X_k \in \mathbb{R}^{n_k \times P}$ as the data matrix in which the rows correspond to the n_k data points assigned to cluster k, and $\boldsymbol{\mu}_k$ as the feature-wise mean vector of X_k. The columns in $V_k^\star = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_P]$ correspond to the eigenvectors of the covariance matrix of X_k, and $\Lambda_k^\star$ is a diagonal matrix containing the set of P eigenvalues $\{\lambda_1, \ldots, \lambda_P\}$. We denote V_k as the subset of eigenvectors in $V_k^\star$ that correspond to the q largest eigenvalues.

Given the subspace bases $\mathcal{V} = \{V_1, \ldots, V_K\}$, the cluster label ω_i for each point $\boldsymbol{x}_i \in \mathcal{X}$ can be obtained as

$$\omega_i = \underset{k \in \{1,\ldots,K\}}{\arg\min} \left\| \boldsymbol{x}_i - V_k V_k^\mathsf{T} \boldsymbol{x}_i \right\|_2^2 . \tag{3.3.5}$$

The algorithm terminates when the loss function value in Eq. (3.3.3) stops decreasing, which indicates that either a local or global optimum is reached.

3.3.2 Query Procedure

The first element in quantifying the influence of an unlabelled point is the reduction in the reconstruction error that would be achieved if this point is removed from its currently assigned cluster. It is important to note that removing a point from a cluster implies that the basis for the associated linear subspace may change, because V_k is a function of X_k (see Eq. (3.3.4)). Explicitly, we define $U_1(\boldsymbol{x}_s, V_{\omega_s})$ as the decrease in the reconstruction error after removing the queried point $\boldsymbol{x}_s$ from cluster ω_s, which can be expressed as

$$U_1(\boldsymbol{x}_s, V_{\omega_s}) = \sum_{\boldsymbol{x} \in \mathcal{X}_{\omega_s}} L(\boldsymbol{x}, V_{\omega_s}) - \sum_{\boldsymbol{x} \in \mathcal{X}_{\omega_s} \setminus \{\boldsymbol{x}_s\}} L(\boldsymbol{x}, \tilde{V}_{\omega_s}), \tag{3.3.6}$$

where $\mathcal{X}_{\omega_s}$ denotes the set of points in cluster ω_s, and V_{ω_s} denotes the basis matrix for cluster ω_s. We use $\tilde{V}_{\omega_s}$ to denote the potentially perturbed basis matrix after point $\boldsymbol{x}_s$ is removed from cluster ω_s.

The second element in quantifying the influence of an unlabelled point is to consider the increase in the reconstruction error of the cluster that $\boldsymbol{x}_s$ will be assigned to (after being removed from its current cluster ω_s). As before, adding a point to a cluster implies that the associated basis for this cluster may change. Given that each point is allocated to its closest subspace, it is thus sensible to assume the cluster that $\boldsymbol{x}_s$ has the second smallest reconstruction error to is where $\boldsymbol{x}_s$ would be assigned next. This can be expressed as

$$\omega_s^\star = \underset{k \in \{1,\ldots,K\} \setminus \{\omega_s\}}{\arg\min} \; L(\boldsymbol{x}_s, V_k). \tag{3.3.7}$$

Then we can define $U_2(\boldsymbol{x}_s, V_{\omega_s^\star})$ as the increase in the reconstruction error after adding $\boldsymbol{x}_s$ to cluster $\omega_s^\star$, which can be expressed as

$$U_2(\boldsymbol{x}_s, V_{\omega_s^\star}) = \sum_{\boldsymbol{x} \in \mathcal{X}_{\omega_s^\star} \cup \{\boldsymbol{x}_s\}} L(\boldsymbol{x}, \tilde{V}_{\omega_s^\star}) - \sum_{\boldsymbol{x} \in \mathcal{X}_{\omega_s^\star}} L(\boldsymbol{x}, V_{\omega_s^\star}). \tag{3.3.8}$$

Here $\tilde{V}_{\omega_s^\star} \in \mathbb{R}^{P \times q}$ is the basis matrix of the points in the set $\left\{\mathcal{X}_{\omega_s^\star} \cup \{\boldsymbol{x}_s\}\right\}$, whose columns correspond to the eigenvectors of its covariance matrix.

Combining the above two influence measures together, we determine the most informative and potentially misclassified point $\boldsymbol{x}_s^\star$ as

$$\boldsymbol{x}_s^\star = \underset{\boldsymbol{x}_s \in \mathcal{X}_U}{\arg\max} \left\{U_1(\boldsymbol{x}_s, V_{\omega_s}) - U_2(\boldsymbol{x}_s, V_{\omega_s^\star})\right\}, \tag{3.3.9}$$

where $\mathcal{X}_U$ denotes the set of unlabelled points. We also denote the set of labelled points as $\mathcal{X}_L$. Eq. (3.3.9) gives the point that brings the largest decrease in the reconstruction error once removed from its allocated cluster ω_s, and the smallest increase in reconstruction error upon being reallocated to its most probable cluster $\omega_s^\star$.

Although these two measures of influence can be quantified and calculated exactly, the number of required SVD computations is $\mathcal{O}(N^2)$ throughout all iterations. Not to mention that the computational complexity of SVD is $\min\{N^2P, P^2N\}$ (Golub and Van Loan, 2013). Every time a point is removed from or added to a cluster, the subspace bases would change and need to be recalculated through PCA. Hence the need to seek for an alternative approach, which could be pursued through the perturbation analysis of PCA (Critchley, 1985).

We approximate the perturbed covariance matrix, the perturbed eigenvectors, and the perturbed eigenvalues through power series expansions (Shi, 1997). As such, we can obtain expressions for the updated reconstruction error without having to recompute all the updated eigenvalues and eigenvectors after data deletion or addition. The algorithmic form of our proposed query strategy is provided in Algorithm 5, before we provide the details of how the two influence measures are calculated.

Algorithm 5: Query Strategy

```
Input :Data matrix: $X \in \mathbb{R}^{N \times P}$
       Number of clusters: $K$
       Initial cluster labels: $\Omega = \{\omega_1, \dots, \omega_N\}$
repeat
    for $\boldsymbol{x}_s \in \mathcal{X}_U$ do
        Compute the influence $U_1(\boldsymbol{x}_s, V_{\omega_s})$ of removing $\boldsymbol{x}_s$ from its allocated
          cluster $\omega_s$
        Calculate $\omega_s^\star = \arg\min_{k \in \{1,\dots,K\} \setminus \{\omega_s\}} L(\boldsymbol{x}_s, V_k)$
        Calculate $U_2(\boldsymbol{x}_s, V_{\omega_s^\star})$ using Eq. (3.3.8)
    end
    Optimise Eq. (3.3.9) to query $\boldsymbol{x}_s^\star$ and its true class $c_s \in \{1, \dots, K\}$
until Budget $T$ or desired performance is reached
```

The influence of data deletion. Let S denote the sample covariance matrix for the points that belong to the same cluster, $\lambda_1, \dots, \lambda_P$ denote its eigenvalues in descending order, and $\boldsymbol{v}_1, \dots, \boldsymbol{v}_P$ denote its eigenvectors. Let $\mathcal{D}$ denote a set of d points to be removed from the cluster. As a result, the covariance matrix, its eigenvectors and eigenvalues will change by a certain amount. Under small perturbations ($0 < \varepsilon < 1$), the perturbed covariance matrix $S(\varepsilon)$, the k-th perturbed eigenvalue $\lambda_k(\varepsilon)$ and the k-th perturbed eigenvector $\boldsymbol{v}_k(\varepsilon)$ can be written as the following convergent power series (Wilkinson, 1965; Bénasséni, 2018):

$$\begin{aligned} S(\varepsilon) &= S + S^{(1)}\varepsilon + S^{(2)}\varepsilon^2 + \cdots + S^{(m)}\varepsilon^m + \cdots , \\ \lambda_k(\varepsilon) &= \lambda_k + \alpha_1\varepsilon + \alpha_2\varepsilon^2 + \cdots + \alpha_m\varepsilon^m + \cdots , \\ \boldsymbol{v}_k(\varepsilon) &= \boldsymbol{v}_k + \boldsymbol{\psi}_1\varepsilon + \boldsymbol{\psi}_2\varepsilon^2 + \cdots + \boldsymbol{\psi}_m\varepsilon^m + \cdots . \end{aligned} \tag{3.3.10}$$

For sufficiently small ε, the order of the eigenvalues is maintained, so are the signs within the eigenvectors (Enguix-González et al., 2005).

The main interest lies in finding the coefficients in the power series approximations. First, the perturbed sample covariance matrix $S^{-}_{(\mathcal{D})}$ can be deduced from the basic definition of a covariance matrix (Wang and Liski, 1993; Bénasséni, 2018),

$$\begin{aligned} S^{-}_{(\mathcal{D})} = S + \frac{d}{n-d}\left((S - S_{\mathcal{D}}) - (\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})^{\mathsf{T}}\right) \\ - \frac{d^2}{(n-d)^2}(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})^{\mathsf{T}}. \end{aligned} \tag{3.3.11}$$

In the above expression, n denotes the original number of points in the cluster that the set of points $\mathcal{D}$ are removed from. We use $\bar{\boldsymbol{x}} \in \mathbb{R}^P$ to denote the feature-wise mean vector of the data before the removal of d points, and $\bar{\boldsymbol{x}}_{\mathcal{D}}$ the feature-wise mean vector of the d points to be removed. Lastly, we use S, $S_{\mathcal{D}}$, and $S^{-}_{(\mathcal{D})}$ to denote the original covariance matrix, the covariance matrix of the set of deleted data points $\mathcal{D}$, and the covariance matrix of the perturbed data, respectively. We can associate $\frac{d}{n-d}$ and $\left((S - S_{\mathcal{D}}) - (\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})^{\mathsf{T}}\right)$ with ε and $S^{(1)}$ as in Eq. (3.3.10). Similarly, the correspondence can be made for the second order coefficients. We use a first order approximation for our purpose from now on, as it has been shown to be sufficiently accurate (Wang and Liski, 1993).

As for the coefficients in the approximations for the eigenvalues and eigenvectors, Lemma 2 in Wang and Liski (1993) provides us with the following results

$$\alpha_1 = \boldsymbol{v}_k^{\mathsf{T}} S^{(1)} \boldsymbol{v}_k, \quad \alpha_m = \boldsymbol{v}_k^{\mathsf{T}} S^{(1)} \boldsymbol{\psi}_{m-1}, \tag{3.3.12}$$

and

$$\begin{aligned}\boldsymbol{\psi}_1 &= -(S-\lambda_k I)^{\dagger} S^{(1)} \boldsymbol{v}_k, \\ \boldsymbol{\psi}_m &= -(S-\lambda_k I)^{\dagger}\left(S^{(1)}\boldsymbol{\psi}_{(m-1)} - \sum_{i=1}^{(m-1)} \alpha_i \boldsymbol{\psi}_{(m-i)}\right).\end{aligned} \tag{3.3.13}$$

In the above expression, we have the Moore-Penrose inverse (Golub and Van Loan, 2013)

$$(S-\lambda_k I)^{\dagger} = \sum_{j\neq k} \frac{\boldsymbol{v}_j \boldsymbol{v}_j^{\mathsf{T}}}{(\lambda_j - \lambda_k)}.$$

Based on the above, we can deduce expressions for the perturbed eigenvalues, and the influence of data deletion as expressed in Eq. (3.3.6).

We start with writing the first order approximation of the k-th ($k \in \{1,\ldots,P\}$) perturbed eigenvalue as follows

$$\begin{aligned}\lambda_k(\varepsilon) &= \lambda_k + \varepsilon\lambda_k^{(1)} + \mathcal{O}(\varepsilon^2) \\ &\approx \lambda_k + \varepsilon \boldsymbol{v}_k^{\mathsf{T}} S^{(1)} \boldsymbol{v}_k \\ &= \lambda_k + \frac{d}{n-d}\boldsymbol{v}_k^{\mathsf{T}}\left[S - S_{\mathcal{D}} - \frac{1}{d}\sum_{s\in\mathcal{D}}(\boldsymbol{x}_s-\bar{\boldsymbol{x}})(\boldsymbol{x}_s-\bar{\boldsymbol{x}})^{\mathsf{T}}\right]\boldsymbol{v}_k \\ &= \lambda_k + \frac{d}{n-d}\left[\lambda_k - \boldsymbol{v}_k^{\mathsf{T}} S_{\mathcal{D}}\boldsymbol{v}_k - \frac{1}{d}\sum_{s\in\mathcal{D}}\alpha_{ks}^2\right] \\ &= \frac{n}{n-d}\lambda_k - \frac{1}{n-d}\sum_{s\in\mathcal{D}}\alpha_{ks}^2 - \frac{d}{n-d}\boldsymbol{v}_k^{\mathsf{T}} S_{\mathcal{D}}\boldsymbol{v}_k,\end{aligned} \tag{3.3.14}$$

where $\alpha_{ks} = \boldsymbol{v}_k^{\mathsf{T}}(\boldsymbol{x}_s - \bar{\boldsymbol{x}})$. Then using the expression in Eq. (3.3.6), we can write the influence of removing a set $\mathcal{D}$ of data $X_{\mathcal{D}} \in \mathbb{R}^{d\times P}$ from cluster k as

$$\begin{aligned}U_1(X_{\mathcal{D}}, V_k) &= \sum_{\boldsymbol{x}\in\mathcal{X}_k} L(\boldsymbol{x}, V_k) - \sum_{\boldsymbol{x}\in\mathcal{X}_k\setminus\{\boldsymbol{x}_s:\, s\in\mathcal{D}\}} L(\boldsymbol{x}, \tilde{V}_k) \\ &= \sum_{k=(q+1)}^{P}\lambda_k - \sum_{k=(q+1)}^{P}\lambda_k(\varepsilon) \\ &= \sum_{k=(q+1)}^{P}\left(\frac{1}{n-d}\sum_{s\in\mathcal{D}}\alpha_{ks}^2 + \frac{d}{n-d}\left(\boldsymbol{v}_k^{\mathsf{T}} S_{\mathcal{D}}\boldsymbol{v}_k - \lambda_k\right)\right).\end{aligned} \tag{3.3.15}$$

One can obtain the influence for the deletion of one point by plugging in $d = 1$. The deduction follows due to the equivalence between the reconstruction error and the sum of the unused eigenvalues in representing the subspace (Jolliffe, 2011). Next, we also need to find another cluster on which the deleted data have little influence if they were added to the cluster.

The influence of data addition. In the previous section, we have shown the influence of data deletion through perturbation analysis of the eigenvalues and eigenvectors. The aim is to find influential points whose true classes might differ from their currently allocated labels.

Now we assess the impact on the reconstruction error for the cluster to which the removed points are added. Following the same line of analysis as before, and with a slight abuse of notation, we now let X denote the data matrix that the set of d points are to be added to, and n the number of points in X. We denote the data after the addition of d points as $X_{\mathcal{D}_+}$, and the corresponding sample covariance matrix $S^+_{(\mathcal{D})}$ which combines the original data X and the data to be added $X_{\mathcal{D}}$.

Proposition 3.3.1. The form of $S^+_{(\mathcal{D})}$ can be expressed as follows,

$$\begin{aligned} S^+_{(\mathcal{D})} = S &+ \frac{d}{n+d}\left((S_{\mathcal{D}} - S) + (\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})^{\mathsf{T}}\right) \\ &- \frac{d^2}{(n+d)^2}\left(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}}\right)\left(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}}\right)^{\mathsf{T}}. \end{aligned} \tag{3.3.16}$$

The proof of Proposition 3.3.1 can be found in Appendix 3.A.1. It is easy to see that this can be matched exactly with the first two orders of the power series expansion. We further show the perturbed form of the covariance matrix for the case of single data addition in Proposition 3.3.2, with the proof included in Appendix 3.A.2. It can be seen that the expression for single data deletion can also be obtained directly by setting $d = 1$ in Eq. (3.3.16).

Proposition 3.3.2. The perturbed covariance matrix in the case when $d = 1$ can be expressed as

$$\begin{aligned} S^{+}_{(s)} &= S + \frac{1}{n+1}\Big((\bar{\boldsymbol{x}} - \boldsymbol{x}_s)(\bar{\boldsymbol{x}} - \boldsymbol{x}_s)^{\mathsf{T}} - S\Big) \\ &\quad - \frac{1}{(n+1)^2}(\bar{\boldsymbol{x}} - \boldsymbol{x}_s)(\bar{\boldsymbol{x}} - \boldsymbol{x}_s)^{\mathsf{T}}, \end{aligned} \tag{3.3.17}$$

in which $\frac{1}{n+1}$ and $\Big[(\bar{\boldsymbol{x}} - \boldsymbol{x}_s)(\bar{\boldsymbol{x}} - \boldsymbol{x}_s)^{\mathsf{T}} - S\Big]$ correspond to ε and $S^{(1)}$ respectively.

Using the above expression for the perturbed covariance matrix and the results in Eq. (3.3.12), we express the first order approximation of the k-th perturbed eigenvalue for $d = 1$ as

$$\begin{aligned} \lambda_k(\varepsilon) &= \lambda_k + \varepsilon\lambda_k^{(1)} + \mathcal{O}(\varepsilon^2) \\ &\approx \lambda_k + \varepsilon \boldsymbol{v}_k^{\mathsf{T}} S^{(1)} \boldsymbol{v}_k \\ &= \lambda_k + \frac{1}{n+1}\boldsymbol{v}_k^{\mathsf{T}}\Big((\bar{\boldsymbol{x}} - \boldsymbol{x}_s)(\bar{\boldsymbol{x}} - \boldsymbol{x}_s)^{\mathsf{T}} - S\Big)\boldsymbol{v}_k \\ &= \frac{1}{n+1}\alpha_{ks}^2 + \frac{n}{n+1}\lambda_k, \end{aligned} \tag{3.3.18}$$

where $\alpha_{ks} = \boldsymbol{v}_k^{\mathsf{T}}(\boldsymbol{x}_s - \bar{\boldsymbol{x}})$ as before. Hence, the change in the reconstruction error for cluster $\omega_s^\star$ after the addition of $\boldsymbol{x}_s$ can be expressed as

$$\begin{aligned} U_2(\boldsymbol{x}_s, V_{\omega_s^\star}) &= \sum_{\boldsymbol{x} \in \mathcal{X}_{\omega_s^\star} \cup \{\boldsymbol{x}_s\}} L(\boldsymbol{x}, \tilde{V}_{\omega_s^\star}) - \sum_{\boldsymbol{x} \in \mathcal{X}_{\omega_s^\star}} L(\boldsymbol{x}, V_{\omega_s^\star}) \\ &= \sum_{k=q+1}^{P} (\lambda_k(\varepsilon) - \lambda_k) \\ &= \sum_{k=q+1}^{P} \frac{\alpha_{ks}^2 - \lambda_k}{n+1}. \end{aligned} \tag{3.3.19}$$

Using the perturbation analysis results, the influence of data addition and deletion can be calculated directly after computing SVD decompositions K times per iteration. This means that we only need to compute SVD decompositions $(T \cdot K)$ times for all T iterations as compared to $\mathcal{O}(T \cdot N^2)$.

3.3.3 Update Procedure

After the class memberships of some points are queried, we will know the pairwise must-link and cannot-link relationships among them. However, we do not know to which cluster label we should assign each of these points to. The next step is to update the subspace model under the grouping constraints. That is, the queried points that belong to the same class must be assigned to the same cluster label. Additionally, the queried points that do not belong to the same class should be assigned to different cluster labels.

We can naturally extend KSC into an iterative constrained clustering algorithm with three stages. The first two stages involve the estimation of subspace bases and the cluster assignment of each point to the closest subspace. In the third stage, we satisfy the grouping constraints as mentioned above. This gives us a new constrained clustering objective, which is composed of two parts.

For the set of unlabelled data $\mathcal{X}_U$, the subspace clustering objective is to minimise

$$L(\mathcal{X}_U, \mathcal{V}) = \sum_{\boldsymbol{x}_u \in \mathcal{X}_U} \left\{ \min_{m \in \{1,\ldots,K\}} \left\| \boldsymbol{x}_u - V_m V_m^{\mathsf{T}} \boldsymbol{x}_u \right\|_2^2 \right\}, \tag{3.3.20}$$

where $V_m \in \mathbb{R}^{P \times q}$ is a basis matrix that is determined by the points that are currently allocated to subspace m. Note that the basis matrix of the m-th cluster V_m is determined by points that are both labelled and unlabelled.

For the set of labelled data $\mathcal{X}_L$, we need to minimise the reconstruction error without violating any of the grouping constraints. Among K groups of queried points, there are $K!$ ways of matching each group to a unique cluster label. This is a combinatorial optimisation problem, and we can denote as $\mathcal{P}(K)$ the set of all possible permutations. Let $P_{n\cdot}$ be the n-th permutation in $\mathcal{P}(K)$ that contains K unique labels to be matched with the queried points, and P_{nc} be the assigned cluster label in the n-th permutation that corresponds to true class c. Then we can write the subspace clustering objective for the labelled data $\mathcal{X}_L$ as

$$L(\mathcal{X}_L, \mathcal{V}) = \min_{\substack{P_{n\cdot} \in \mathcal{P}(K) \\ n \in \{1,\ldots,K!\}}} \left\{ \sum_{c=1}^{K} \left\| X_c - V_{P_{nc}} V_{P_{nc}}^{\mathsf{T}} X_c \right\|_2^2 \right\}, \tag{3.3.21}$$

where X_c is a $n_c \times P$ matrix that contains the n_c queried points from class c.

When K is small, it is easy to simply evaluate all $K!$ permutations and choose the one with the smallest overall cost. However, as the number of clusters grows, it is computationally prohibitive to evaluate all combinatorial possibilities. This problem is also known as the *minimum weight perfect matching problem*, which can be solved in polynomial time through the *Hungarian algorithm* (Kuhn, 1955). We first construct a K by K cost matrix P in which the (i, j)-th entry P_{ij} denotes the total reconstruction error of allocating data from class i to cluster label j. An improved variant of the Hungarian algorithm can achieve a computational cost of $\mathcal{O}(K^3)$ (Jonker and Volgenant, 1987). Hence, we adopt it as an alternative approach to exhaustive search to our problem in stage 3 when $K!$ is larger than K^3.

To combine both the unlabelled and labelled objectives together, we can express the combined constrained objective function as

$$\begin{aligned} g(\mathcal{X}, \mathcal{V}) = \sum_{\boldsymbol{x}_u \in \mathcal{X}_U} &\left\{ \min_{m \in \{1,\ldots,K\}} \left\| \boldsymbol{x}_u - V_m V_m^{\mathsf{T}} \boldsymbol{x}_u \right\|_2^2 \right\} + \\ &\min_{\substack{P_{n\cdot} \in \mathcal{P}(K), \\ n \in \{1,\ldots,K!\}}} \left\{ \sum_{c=1}^{K} \left\| X_c - V_{P_{nc}} V_{P_{nc}}^{\mathsf{T}} X_c \right\|_2^2 \right\}. \end{aligned} \tag{3.3.22}$$

The procedural form of *KSC with Constraints (KSCC)* is detailed in Algorithm 6.

This three-stage procedure ensures that the constrained subspace clustering objective in Eq. (3.3.22) decreases monotonically whilst satisfying all of the grouping constraints. A proof for this can be found in Theorem 3.3.1.

Theorem 3.3.1. The *K-Subspace Clustering with Constraints (KSCC)* algorithm decreases the objective in Eq. (3.3.22) monotonically throughout iterations.

Proof. Our proof borrows ideas from the proof for the monotonicity of the K-means clustering algorithm. For initialisation, we have as input a set of cluster labels $\Omega^{(0)} = \left\{\omega_1^{(0)}, \ldots, \omega_N^{(0)}\right\}$, the set of unlabelled data $\mathcal{X}_U$, and the set of labelled data $\mathcal{X}_L$. Given the input information, we can calculate an initial set of bases matrices $\mathcal{V}^{(0)}$ for all subspaces. Let $g(\mathcal{X}, \mathcal{V}^{(0)})$ be the initial combined reconstruction error, then at iteration

Algorithm 6: KSC with Constraints (KSCC)

```
Input :Labelled and unlabelled data: $\mathcal{X}_L$, $\mathcal{X}_U$
       Initial cluster labels: $\Omega^{(0)} = \left\{\omega_1^{(0)}, \dots, \omega_N^{(0)}\right\}$
       Subspace dimension: $q$
repeat
    % Stage 1: fitting subspaces
    for $X_k \in \mathcal{X}$ ($k = 1, \dots, K$) do
        Calculate the eigen-decomposition on the covariance matrix of $X_k$:
        $\mathrm{cov}(X_k) = V_k \Lambda_k V_k^\mathsf{T}$
    end
    % Stage 2: updating labels
    for $\boldsymbol{x}_i \in \mathcal{X}$ ($i = 1, \dots, N$) do
        Determine the cluster label for $\boldsymbol{x}_i$:
        $\omega_i = \arg\min_{\omega_i \in \{1,\dots,K\}} \left\|\boldsymbol{x}_i - V_k V_k^\mathsf{T} \boldsymbol{x}_i\right\|_2^2$
    end
    % Stage 3: satisfying constraints
    Find the best vector $P_{n\cdot}^\star \in \mathcal{P}(K)$ to match with the true classes by solving
      Eq. (3.3.21):
        $$P_{n\cdot}^\star = \mathop{\arg\min}_{\substack{P_{n\cdot} \in \mathcal{P}(K), \\ n \in \{1,\dots,K!\}}} \left\{ \sum_{c=1}^{K} \left\|X_c - V_{P_{nc}} V_{P_{nc}}^\mathsf{T} X_c\right\|_2^2 \right\}$$
      using the Hungarian algorithm
until Iteration number $T$ is reached or the total reconstruction error stops
  decreasing
```

t ($t = 0, \dots, T$) we have

$$\begin{aligned}
g(\mathcal{X}, \mathcal{V}^{(t)}) &= \sum_{\boldsymbol{x}_u \in \mathcal{X}_U} \left\|\boldsymbol{x}_u - V^{(t)}_{\omega_u^{(t)}} [V^{(t)}_{\omega_u^{(t)}}]^\mathsf{T} \boldsymbol{x}_u\right\|_2^2 + \sum_{c=1}^{K} \left\|X_c - V^{(t)}_{P_{nc}^{(t)}} [V^{(t)}_{P_{nc}^{(t)}}]^\mathsf{T} X_c\right\|_2^2 \\
&\geqslant \sum_{\boldsymbol{x}_i \in \mathcal{X}} \left\|\boldsymbol{x}_i - V^{(t+1)}_{\omega_i^{(t)}} [V^{(t+1)}_{\omega_i^{(t)}}]^\mathsf{T} \boldsymbol{x}_i\right\|_2^2 \\
&= \sum_{\boldsymbol{x}_u \in \mathcal{X}_U} \left\|\boldsymbol{x}_u - V^{(t+1)}_{\omega_u^{(t)}} [V^{(t+1)}_{\omega_u^{(t)}}]^\mathsf{T} \boldsymbol{x}_u\right\|_2^2 + \sum_{c=1}^{K} \left\|X_c - V^{(t+1)}_{P_{nc}^{(t)}} [V^{(t+1)}_{P_{nc}^{(t)}}]^\mathsf{T} X_c\right\|_2^2 \\
&\geqslant \sum_{\boldsymbol{x}_u \in \mathcal{X}_U} \left\|\boldsymbol{x}_u - V^{(t+1)}_{\omega_u^{(t+1)}} [V^{(t+1)}_{\omega_u^{(t+1)}}]^\mathsf{T} \boldsymbol{x}_u\right\|_2^2 + \sum_{c=1}^{K} \left\|X_c - V^{(t+1)}_{P_{nc}^{(t)}} [V^{(t+1)}_{P_{nc}^{(t)}}]^\mathsf{T} X_c\right\|_2^2 \\
&\geq \sum_{\boldsymbol{x}_u \in \mathcal{X}_U} \left\|\boldsymbol{x}_u - V^{(t+1)}_{\omega_u^{(t+1)}} [V^{(t+1)}_{\omega_u^{(t+1)}}]^\mathsf{T} \boldsymbol{x}_u\right\|_2^2 + \sum_{c=1}^{K} \left\|X_c - V^{(t+1)}_{P_{nc}^{(t+1)}} [V^{(t+1)}_{P_{nc}^{(t+1)}}]^\mathsf{T} X_c\right\|_2^2 \\
&= g(\mathcal{X}, \mathcal{V}^{(t+1)}).
\end{aligned}$$

$\square$

The first line of the proof says that, at iteration t, we have a set of cluster labels for the unlabelled data $\Omega_U^{(t)}$ and for the labelled data $\Omega_L^{(t)}$ that satisfy all constraints imposed upon knowing the true classes of the points in $\mathcal{X}_L$. When we proceed into the next step of updating the set of bases $\mathcal{V}^{(t+1)}$ at iteration $(t+1)$, the new set of bases minimise the reconstruction error within each cluster of points given the assignment $\Omega^{(t)}$ (as stated in the second and third lines of the proof). Next, in the assignment update stage for the unlabelled data, we obtain the fourth line of the proof. It states that the assignment $\omega_u^{(t+1)}$ in the $(t+1)$-th iteration would only be different from $\omega_u^{(t)}$ if it gives a smaller reconstruction error for $\boldsymbol{x}_u$. Finally, in the last step of the KSCC algorithm, we update the matching between the cluster labels and the true classes. It only gets updated if some other matching has a smaller overall reconstruction error for the labelled data $\mathcal{X}_L$, which is reflected in the last two lines of the proof.

3.4 Experimental Results

In this section, we conduct a series of experiments with both synthetic and real data to evaluate the performance of our proposed active learning strategies against three other competing strategies[1]. The cluster performance is measured by the Normalised Mutual Information (NMI) (Cover and Thomas, 2012).

In order to inspect the influence of data addition and deletion separately, we use three versions of our proposed active learning strategy: *SCAL-A* and *SCAL-D* only take into account the influence of data addition and data deletion respectively, and *SCAL* is the combined strategy that takes into account both. We compare the performance of our proposed active learning strategies with three alternative schemes: *MaxResid*, *MinMargin* (Lipor and Balzano, 2015), and *Random* strategy. *MaxResid* selects data points that have the largest reconstruction error to their corresponding subspaces. *MinMargin* selects data points that are most equidistant to their two closest subspaces. Lastly as a benchmark,

[1]The code is available at: `https://github.com/hankuipeng/SCAL`.

we compare to random sampling and satisfy the constraints using the KSCC algorithm.

3.4.1 Synthetic Data

For all synthetic experiments, we initialise the cluster labels with the best initialisation (the one with the lowest reconstruction error) out of 50 runs of the KSC algorithm. We set the total number of iterations T of the active learning procedure to be N, and query one point at a time.

Experiment 1: Varying noise level (σ). We investigate the effectiveness of our proposed active learning strategy under varying levels of additive noise. The effectiveness of various strategies are also compared under various levels of additive noise. The data corrupted by noise can be expressed as $Y = X + E$, where X is the noise-free data and E the noise component.

In order to generate the noise-free data matrix $X \in \mathbb{R}^{N\times P}$, we need to generate each sub-matrix $X_k \in \mathbb{R}^{n_k\times P}$ ($k \in \{1,\ldots,K\}$) for each subspace individually and concatenate them to form X. For each subspace $\mathcal{S}_k$, we first generate a $P \times q$ matrix $B_k^\star$ whose entries come from the standard Normal distribution with $\mathcal{N}(0,1)$. Then we orthogonalise the columns of $B_k^\star$ to obtain the matrix B_k whose columns correspond to the basis vectors for the subspace. The noise-free sub-matrix X_k can thus be obtained as

$$X_k = (B_k C_k)^\mathsf{T}, \tag{3.4.1}$$

where $C_k \in \mathbb{R}^{q\times n_k}$ is the coefficient matrix, whose entries are also sampled from the standard Normal distribution. Each column of C_k corresponds to the coefficient vector of a point along the q subspace dimensions. Each entry in the noise data matrix E is generated from standard Normal distribution $\mathcal{N}(0,\sigma^2)$, with zero mean and variance σ^2. We specify additive noise levels to be $\sigma = 0.2$, 0.4, and 0.6 respectively. Across all noise levels, there are 5 clusters in each data set and each cluster contains 200 points from the same subspace of dimension 10 out of the full dimension 20.

The performance of various strategies under all settings are shown in Table 3.1. In general, all strategies require a big proportion of points to be queried to reach perfection as the noise level goes up. It seems most of the advantage of *SCAL* comes from the influence of data addition *SCAL-A*, and it is difficult to say whether there is a difference between *SCAL* and *SCAL-A*. It is worth noting that *MinMargin* has similar performance to *SCAL* and *SCAL-A* when $\sigma = 0.2$.

Experiment 2: Varying angles between subspaces (θ). In order to fix a vector to rotate the subspaces, we apply various active learning strategies on three 3-dimensional examples with subspace dimension $q = 2$. 600 points are generated in total from 3 clusters, and every cluster contains 200 points each. Noise with $\sigma = 0.1$ is added to the data, and the between-subspace angle is specified to be 30, 50, and 70 degrees respectively.

The performance results under all scenarios are shown in Table 3.1. Our proposed active strategy *SCAL* and *SCAL-A* outperform all other strategies significantly. For these two strategies, the proportion of data needed in order to achieve perfection decreases as the between-subspace angle increases. Other strategies have to query almost all points in order to achieve perfect performance apart from *MinMargin*, which is our close competitor in the varying noise setting.

Parameters	SCAL	SCAL-A	SCAL-D	MaxResid	MinMargin	Random
$\sigma = 0.2$	**0.30%**	0.40%	45.20%	19.20%	0.70%	23.00%
$\sigma = 0.4$	**43.10%**	46.10%	99.00%	98.00%	83.10%	99.50%
$\sigma = 0.6$	85.60%	**85.40%**	99.90%	99.10%	89.50%	99.50%
$\theta = 30$	**41.67%**	44.17%	98.67%	99.83%	96.00%	99.00%
$\theta = 50$	37.17%	**36.83%**	99.00%	98.17%	69.50%	99.50%
$\theta = 70$	32.17%	**31.83%**	98.83%	98.50%	77.67%	99.83%

Table 3.1: The percentage of points queried before perfect cluster performance (as evaluated by NMI) is reached on synthetic data sets.

Again, *SCAL-D* strategy as part of our proposed *SCAL* strategy barely distinguishes itself from *Random* strategy. This is within our expectations for two reasons. First, misclassified points are most likely to belong to the nearest cluster that they have the second least reconstruction error to. Secondly, those points whose deletion has a large influence on their allocated subspaces are likely to be correctly classified in the first place due to the level of noise in the data.

3.4.2 Real Data

In this section, we conduct experiments on real-world data comparing *SCAL* to various competing strategies. We demonstrate the advantage of our proposed active learning strategy when the data exhibit subspace structure. Specifically, we experiment with data sets in motion segmentation and face clustering which have been used previously to demonstrate the effectiveness of subspace clustering (Elhamifar and Vidal, 2013).

For KSC-based experiments, we experiment with two initialisation schemes. First, we initialise with the output given by KSC, which is the set of labels that gives the smallest reconstruction error out of 50 runs each with randomly allocated initial labels. Secondly, we initialise with the output from Sparse Subspace Clustering (SSC) (Elhamifar and Vidal, 2013) under the default model parameters. Due to the excellent performance of SSC, the aim is to investigate whether the correct initialisation of subspace bases would help accelerate the performance improvement.

All results are presented in two measures: the first row presents the percentage of data that needs to be queried before perfect clustering is reached; the second row presents the percentage of the area under the plotted performance improvement curve over the total area. The first measure focuses on the amount of queries needed to reach perfect cluster performance, whereas the second measure reflects the overall effectiveness of a query strategy.

Motion segmentation. In this set of experiments, we evaluate the performance of all strategies on six motion segmentation data sets (Tron and Vidal, 2007). Motion segmentation refers to the problem of separating the points in a sequence of frames that compose one video after being combined consecutively. Each point can be represented by a $2f$-dimensional vector, in which f is the number of frames in the video (Elhamifar and Vidal, 2013). Following the parameter setting from Elhamifar and Vidal (2013), we set the subspace dimension $q = 3$, and query one point at each iteration. The performance results are summarised in Table 3.2, and the number of moving objects (clusters) for each data application is also included in the table.

KSC update (KSC initialisation)				
	SCAL	*MinMargin*	*MaxResid*	*Random*
truck2	**4.53%**	90.63%	99.70%	91.84%
$(K = 2)$	**99.36%**	95.46%	86.60%	89.75%
kanatani3	**26.03%**	31.51%	86.30%	91.78%
$(K = 2)$	**86.05%**	84.86%	72.10%	53.10%
1R2TCR	**26.98%**	43.35%	95.68%	88.13%
$(K = 3)$	94.89%	**95.96%**	85.44%	83.12%
three-cars	**31.21%**	60.12%	99.42%	72.25%
$(K = 3)$	**94.18%**	89.07%	86.73%	87.16%
2R3RTC	**37.28%**	52.51%	46.49%	99.20%
$(K = 3)$	96.68%	**97.23%**	95.37%	87.86%
two-cranes	**71.28%**	81.92%	89.36%	97.87%
$(K = 3)$	**59.52%**	58.09%	53.09%	58.49%
KSC update (SSC initialisation)				
1R2TCR	**26.98%**	43.35%	95.68%	98.02%
	94.89%	**95.96%**	85.44%	83.09%
three-cars	**1.73%**	83.82%	99.42%	84.39%
	99.94%	97.61%	95.52%	97.49%
2R3RTC	**9.82%**	40.88%	59.92%	78.96%
	99.76%	99.27%	98.21%	97.73%
two-cranes	**73.40%**	75.53%	98.94%	98.94%
	64.38%	**71.23%**	45.72%	65.80%

Table 3.2: Cluster performance of various active learning strategies on motion segmentation data sets.

It is worth noting that SSC achieves perfect performance on 'truck2' and 'kanatani3', thus there is no need for active learning. The performance improvement over iterations is shown in Figure 3.4.1. We see that the performance of *MinMargin* is very similar to that of *SCAL* most of the time. This is also reflected in the second row of the performance of each data set in Table 3.2. However, *SCAL* always achieves perfect cluster performance first, which is what we expect to see due to its ability to query potentially misclassified points that are also informative. The performance of *MaxResid* also improves rapidly in most scenarios, but it struggles to query the points that lead to perfect performance.

Face clustering. The original Extended Yale Face Database B (Lee et al., 2005) consists of 64 images of 38 distinct faces under various lighting conditions. Each original image is of size 192×168, and have been downsampled to have size 48×42 (Elhamifar

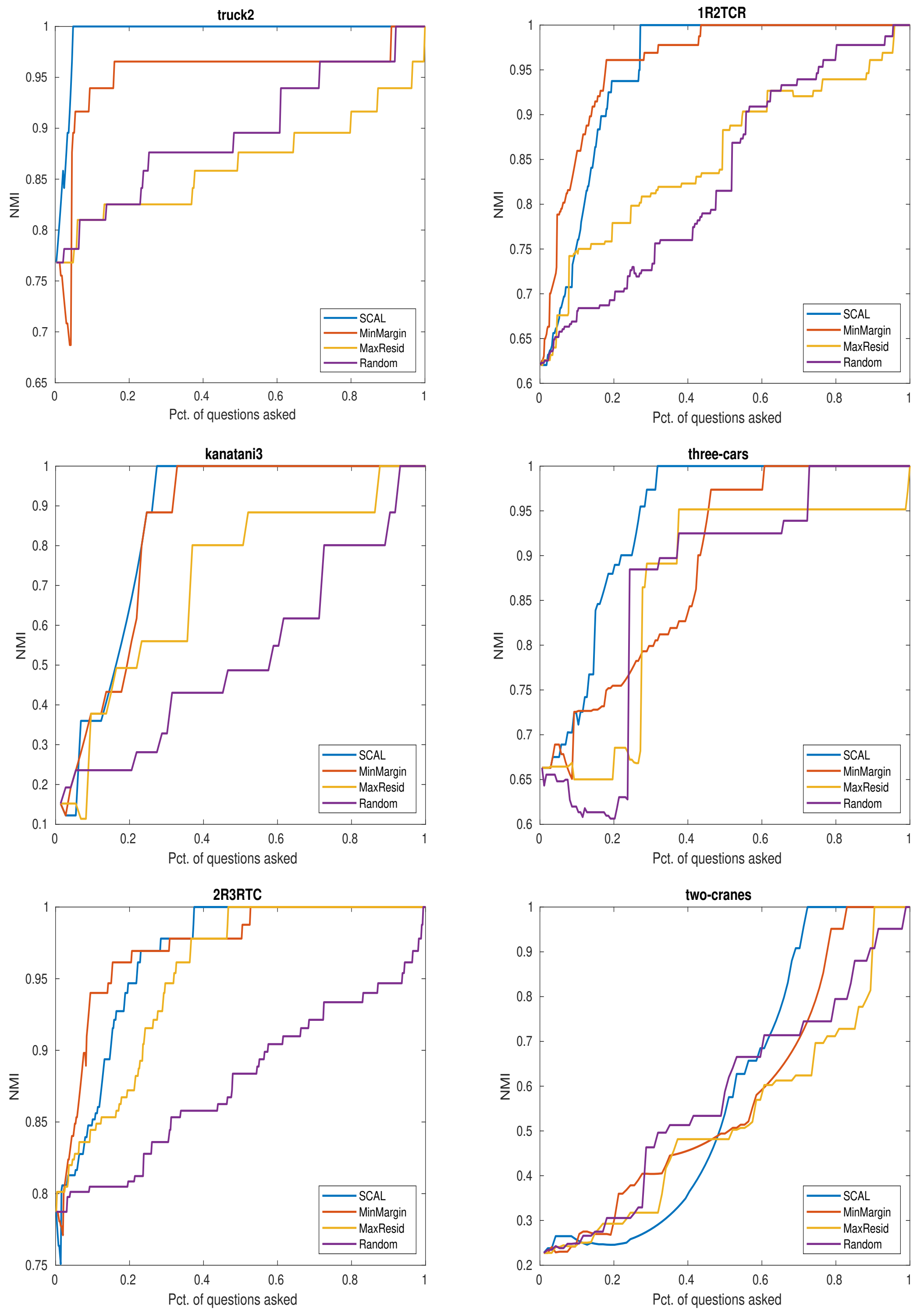


Figure 3.4.1: Performance results measured by NMI on six motion segmentation data sets with KSC initialisation.

and Vidal, 2013). It has previously been shown that under the Lambertian assumption, images of a subject lie close to a linear subspace of dimension 9 (Basri and Jacobs, 2003). Since the data are intrinsically low-dimensional, we preprocess the data by projecting onto its first $5K$ principal components as has been done in (Balcan et al., 2007). Following the experimental settings in Elhamifar and Vidal (2013), we experiment with $K = 2$, 3, 5, 8, and 10. The corresponding data sets are obtained from the SSC package in MATLAB (Elhamifar and Vidal, 2013).

As before, we apply all active learning strategies with both KSC and SSC initialisations. It is worth noting that SSC achieves perfect performance on the preprocessed data when $K = 2$, thus there is no need for active learning. From the results shown in Table 3.3, we see that the percentage of data that needs to be queried goes up with the increase of K. Although the proportion of area under the curve is very similar between *MinMargin* and *SCAL*, *MinMargin* requires a much higher percentage of queries than *SCAL* before perfect clustering is reached.

The performance improvement over time with KSC initialisation is shown in Figure 3.4.2. The initial performance decreases slowly as K increases, and the performance of various active strategies gets closer. With that said, the performance of *SCAL* and *MinMargin* still stand out from the rest.

3.5 Extension to Spectral Clustering

Finally, we make an initial attempt to extend our active learning framework to the spectral clustering setting. A large number of subspace clustering algorithms are spectral-based methods. These methods construct a pairwise affinity matrix through various optimisation schemes and solve the cluster assignment problem through spectral clustering (Elhamifar and Vidal, 2013; Liu et al., 2010). We incorporate the queried information in the similarity matrix and compare with other strategies in the spectral setting.

Using our proposed strategy, the points are queried in the same manner as before. Upon receiving the class information of some points, the constraints are satisfied by updating the affinity matrix. Following the update procedure in Lipor and Balzano

KSC update (KSC initialisation)					
Strategies	$K = 2$	$K = 3$	$K = 5$	$K = 8$	$K = 10$
SCAL	**21.09%**	**19.79%**	**24.06%**	**63.48%**	**53.91%**
	96.10%	**98.28%**	**93.80%**	80.02%	86.45%
MinMargin	64.84%	36.46%	83.13%	98.44%	99.84%
	90.30%	97.31%	91.83%	**81.92%**	**88.95%**
MaxResid	96.88%	95.31%	99.69%	99.81%	99.84%
	77.19%	85.85%	77.43%	79.01%	82.22%
Random	97.66%	96.35%	99.69%	97.85%	97.97%
	77.59%	88.08%	76.27%	77.77%	83.46%

KSC update (SSC initialisation)				
Strategies	$K = 3$	$K = 5$	$K = 8$	$K = 10$
SCAL	**10.94%**	**25.31%**	**28.52%**	**58.91%**
	99.06%	**99.09%**	**98.24%**	95.15%
MinMargin	16.15%	38.13%	92.58%	62.18%
	98.59%	98.14%	98.00%	**97.15%**
MaxResid	93.23%	99.69%	99.81%	99.38%
	91.50%	82.66%	89.75%	93.34%
Random	91.15%	76.25%	99.02%	91.88%
	92.19%	94.40%	91.59%	94.75%

Spectral update (SSC initialisation)				
Strategies	$K = 3$	$K = 5$	$K = 8$	$K = 10$
SCAL	**10.94%**	**26.56%**	**40.62%**	26.56%
	98.91%	**96.03%**	**91.08%**	98.24%
SUPERPAC	14.06%	31.25%	**40.62%**	**20.31%**
	98.39%	95.96%	90.03%	98.67%
MaxResid	90.62%	98.44%	98.44%	57.81%
	93.23%	92.84%	86.81%	**98.76%**
Random	95.31%	98.44%	92.19%	96.88%
	90.90%	92.65%	87.50%	96.06%

Table 3.3: Cluster performance on Yale Faces data sets.

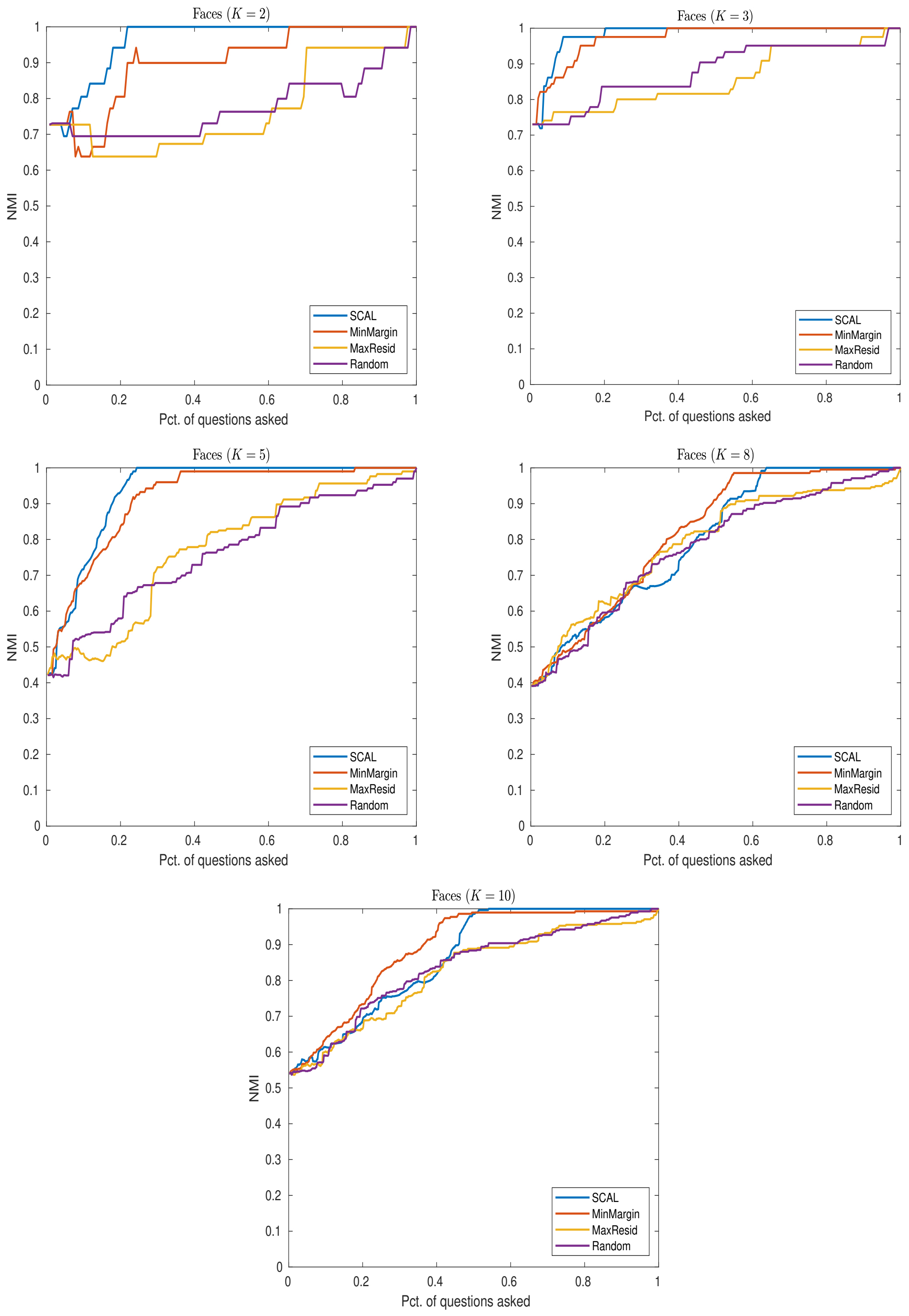


Figure 3.4.2: Performance results measured by NMI on Yale Faces data sets with KSC initialisation.

(2017), we set to ones for those labelled data that belong to the same class and zeros for those that lie in different classes. Spectral clustering is then applied to the updated affinity matrix to obtain labels for all points. As a final step, we apply KSCC to ensure that all grouping constraints are satisfied for the labelled data.

The performance results are shown in the last section of Table 3.3. Note that the authors that propose *MinMargin* have renamed it to *SUPERPAC* for the spectral setting. *SCAL* outperforms other competing strategies in all scenarios apart from when $K = 10$. With that said, *SCAL* still enjoys the same level of rate of improvement as *SUPERPAC* and *MaxResid* when $K = 10$.

3.6 Conclusions & Future Work

We proposed a novel active learning framework for subspace clustering. Ideas from matrix perturbation theory are borrowed to enable efficient estimation of the influence of data deletion and data addition as measured by the change in the reconstruction error. New results on the perturbation analysis of data addition are provided as a by-product of our proposed active learning framework. In addition, we propose a constrained subspace clustering algorithm called K-Subspace Clustering with Constraints (KSCC) that monotonically decreases the constrained objective over iterations.

For future research, there are a few interesting directions we would like to pursue. Firstly, we would like to extend our framework to be able to incorporate not only class information, but also ‘must-link’ and ‘cannot-link’ constraints. Secondly, we would like to consider extensions of our proposed framework in the spectral-based setting. Our initial experimental results seem promising on the Faces data sets using the straightforward spectral update on the affinity matrix. Finally, it would be interesting to explore the scenario where multiple labellers exist, and the provided labels do not necessarily agree with each other.

3.A Appendix

3.A.1 Proof of Proposition 1

In this section, we provide the proof for Proposition 3.3.1 regarding the perturbed form of the covariance matrix after data addition.

Proposition 1. The form of $S^{+}_{(\mathcal{D})}$ can be expressed as,

$$S^{+}_{(\mathcal{D})} = S + \frac{d}{n+d}\left((S_{\mathcal{D}} - S) + (\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})^{\mathsf{T}}\right) - \frac{d^2}{(n+d)^2}\left(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}}\right)\left(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}}\right)^{\mathsf{T}}. \tag{3.A.1}$$

Proof. The sample covariance matrices S, $S_{\mathcal{D}}$, and $S^{+}_{(\mathcal{D})}$ are given as

$$\begin{aligned}
S &= \frac{1}{n}X^{\mathsf{T}}\left(I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^{\mathsf{T}}\right)X,\\
S_{\mathcal{D}} &= \frac{1}{d}X_{\mathcal{D}}^{\mathsf{T}}\left(I_d - \frac{1}{d}\mathbf{1}_d\mathbf{1}_d^{\mathsf{T}}\right)X_{\mathcal{D}},\\
S^{+}_{(\mathcal{D})} &= \frac{1}{n+d}X_{\mathcal{D}_+}^{\mathsf{T}}\left(I_{(n+d)} - \frac{1}{n+d}\mathbf{1}_{(n+d)}\mathbf{1}_{(n+d)}^{\mathsf{T}}\right)X_{\mathcal{D}_+},
\end{aligned} \tag{3.A.2}$$

in which I_n is an identity matrix of size $n \times n$ and $\mathbf{1}_n$ is a vector of all ones with length n. Starting from the expression for $S^{+}_{(\mathcal{D})}$ above, we can write:

$$\begin{aligned}
(n+d)S^{+}_{(\mathcal{D})} &= X_{\mathcal{D}_+}^{\mathsf{T}}X_{\mathcal{D}_+} - \frac{1}{n+d}X_{\mathcal{D}_+}^{\mathsf{T}}\mathbf{1}_{(n+d)}\mathbf{1}_{(n+d)}^{\mathsf{T}}X_{\mathcal{D}_+}\\
&= X^{\mathsf{T}}X + X_{\mathcal{D}}^{\mathsf{T}}X_{\mathcal{D}} - (n+d)\bar{\boldsymbol{x}}_{\mathcal{D}}\bar{\boldsymbol{x}}_{\mathcal{D}}^{\mathsf{T}}\\
&= X^{\mathsf{T}}X + X_{\mathcal{D}}^{\mathsf{T}}X_{\mathcal{D}} - \frac{1}{n+d}(n\bar{\boldsymbol{x}} + d\bar{\boldsymbol{x}}_{\mathcal{D}})(n\bar{\boldsymbol{x}} + d\bar{\boldsymbol{x}}_{\mathcal{D}})^{\mathsf{T}}\\
&= nS + dS_{\mathcal{D}} + \frac{1}{n}X^{\mathsf{T}}\mathbf{1}_n\mathbf{1}_n^{\mathsf{T}}X + \frac{1}{d}X_{\mathcal{D}}^{\mathsf{T}}\mathbf{1}_d\mathbf{1}_d^{\mathsf{T}}X_{\mathcal{D}} - \frac{1}{n+d}(n\bar{\boldsymbol{x}} + d\bar{\boldsymbol{x}}_{\mathcal{D}})(n\bar{\boldsymbol{x}} + d\bar{\boldsymbol{x}}_{\mathcal{D}})^{\mathsf{T}}\\
&= nS + dS_{\mathcal{D}} + n\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^{\mathsf{T}} + d\bar{\boldsymbol{x}}_{\mathcal{D}}\bar{\boldsymbol{x}}_{\mathcal{D}}^{\mathsf{T}} - \frac{nd}{n+d}\left(\frac{n}{d}\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^{\mathsf{T}} + \bar{\boldsymbol{x}}\bar{\boldsymbol{x}}_{\mathcal{D}}^{\mathsf{T}} + \bar{\boldsymbol{x}}_{\mathcal{D}}\bar{\boldsymbol{x}}^{\mathsf{T}} + \frac{d}{n}\bar{\boldsymbol{x}}_{\mathcal{D}}\bar{\boldsymbol{x}}_{\mathcal{D}}^{\mathsf{T}}\right)\\
&= nS + dS_{\mathcal{D}} + \frac{nd}{n+d}\left(\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^{\mathsf{T}} - 2\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}_{\mathcal{D}}^{\mathsf{T}} + \bar{\boldsymbol{x}}_{\mathcal{D}}\bar{\boldsymbol{x}}_{\mathcal{D}}^{\mathsf{T}}\right)\\
&= nS + dS_{\mathcal{D}} + \frac{nd}{n+d}(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})^{\mathsf{T}}\\
&= (n+d)S + d\left(S_{\mathcal{D}} - S\right) + d(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})^{\mathsf{T}} - \frac{d^2}{n+d}(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_{\mathcal{D}} - \bar{\boldsymbol{x}})^{\mathsf{T}},
\end{aligned}$$

from which the result follows. □

3.A.2 Proof of Proposition 2

In this section, we provide the proof for Proposition 3.3.2 regarding the perturbed form of the covariance matrix after the addition of one single point.

Proposition 2. Following the same line of analysis, we can show that the perturbed covariance matrix in the case when $d = 1$ can be expressed as,

$$S^+_{(s)} = S + \frac{1}{n+1}\Big(\left(\bar{\boldsymbol{x}} - \boldsymbol{x}_s\right)\left(\bar{\boldsymbol{x}} - \boldsymbol{x}_s\right)^\mathsf{T} - S\Big) - \frac{1}{(n+1)^2}\left(\bar{\boldsymbol{x}} - \boldsymbol{x}_s\right)\left(\bar{\boldsymbol{x}} - \boldsymbol{x}_s\right)^\mathsf{T}, \tag{3.A.3}$$

in which we can think of $\frac{1}{n+1}$ and $\left[\left(\bar{\boldsymbol{x}} - \boldsymbol{x}_s\right)\left(\bar{\boldsymbol{x}} - \boldsymbol{x}_s\right)^\mathsf{T} - S\right]$ as ε and $S^{(1)}$ respectively.

Proof.

$$\begin{aligned}
(n+1)S^+_{(s)} &= X^\mathsf{T}_{s+}X_{s+} - \frac{1}{n+1}X^\mathsf{T}_{s+}\mathbf{1}_{(n+1)}\mathbf{1}^\mathsf{T}_{(n+1)}X_{s+} \\
&= X^\mathsf{T}X + \boldsymbol{x}_s\boldsymbol{x}_s^\mathsf{T} - \frac{1}{n+1}(n\bar{\boldsymbol{x}} + \boldsymbol{x}_s)(n\bar{\boldsymbol{x}} + \boldsymbol{x}_s)^\mathsf{T} \\
&= nS + \frac{1}{n}X^\mathsf{T}\mathbf{1}_n\mathbf{1}_n^\mathsf{T}X + \boldsymbol{x}_s\boldsymbol{x}_s^\mathsf{T} - \frac{1}{n+1}\left(n^2\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^\mathsf{T} + 2n\bar{\boldsymbol{x}}\boldsymbol{x}_s^\mathsf{T} + \boldsymbol{x}_s\boldsymbol{x}_s^\mathsf{T}\right) \\
&= nS + n\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^\mathsf{T} - \frac{1}{n+1}\left(n^2\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^\mathsf{T} + 2n\bar{\boldsymbol{x}}\boldsymbol{x}_s^\mathsf{T} - n\boldsymbol{x}_s\boldsymbol{x}_s^\mathsf{T}\right) \\
&= nS + \frac{n}{n+1}\left(\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^\mathsf{T} - 2\bar{\boldsymbol{x}}\boldsymbol{x}_s^\mathsf{T} + \boldsymbol{x}_s\boldsymbol{x}_s^\mathsf{T}\right) \\
&= nS + \frac{n}{n+1}\left(\bar{\boldsymbol{x}} - \boldsymbol{x}_s\right)\left(\bar{\boldsymbol{x}} - \boldsymbol{x}_s\right)^\mathsf{T} \\
&= nS + \left(\bar{\boldsymbol{x}} - \boldsymbol{x}_s\right)\left(\bar{\boldsymbol{x}} - \boldsymbol{x}_s\right)^\mathsf{T} - \frac{1}{n+1}\left(\bar{\boldsymbol{x}} - \boldsymbol{x}_s\right)\left(\bar{\boldsymbol{x}} - \boldsymbol{x}_s\right)^\mathsf{T}.
\end{aligned}$$

□

Chapter 4

Weighted Sparse Simplex Representation

Subspace clustering methods that express each point as a linear combination of other points have achieved great success in many real world applications, for example gene sequencing, image recognition, and motion segmentation. In real-world applications, it is not always easy to validate the cluster performance of these methods due to the scarcity of available labelling information and the cost of obtaining them. Existing literature addresses the problem of subspace clustering, the problem of obtaining useful labels – active learning, and the problem of incorporating available labels – constrained clustering, separately. In this work, we build a unified framework for spectral-based subspace clustering and active learning.

The initial stage of this framework is to obtain a data partitioning. We develop a spectral-based subspace clustering method named Weighted Sparse Simplex Representation (WSSR), which uses local neighbourhood information to represent each point as a sparse linear combination of other points. Given a data partitioning, the main framework is composed of two stages. In the first stage, we query the labels for the most informative points given the current subspace structure. In the second stage, we propose an extension to WSSR, named WSSR+, that incorporates available labelling information into the WSSR problem formulation. Experiments on both synthetic and real data demonstrate the effectiveness of our proposed framework.

Keywords: Subspace clustering; Constrained clustering; Active learning.

4.1 Introduction

Data have been generated at unprecedented speed and quantity in recent years. High-dimensional data, in particular, are ubiquitous in numerous application domains. For example in genomics, microarray technologies provide high-dimensional gene expression measurements that are used to identify sub-types of cancer (McWilliams and Montana, 2014). Recent technological advances have enabled people to take high-quality photos with millions of pixels. This gives rise to problems such as image representation (Hong et al., 2006), and motion segmentation (Rao et al., 2010) in computer vision. Due to the rapid development of Natural Language Processing (NLP) in the past few years, many sophisticated language models have been developed to represent text data. These representations often come at hundreds and even thousands of dimensions (Devlin et al., 2018; Yang et al., 2019b; Wang and Kuo, 2020).

In these applications where high dimensional data are abundant, it is often the case that the main structure of the data can be well represented in much lower dimensional subspaces. When the goal is to identify one common low dimensional structure in the data, classical methods such as Principal Component Analysis (PCA) (Jolliffe, 2011) are often used for dimension reduction. However more advanced techniques are needed if there are multiple subspace structures in the data, where each structure represents data points from a distinct group. The problem of simultaneously estimating the corresponding subspace structure for each cluster and partitioning a group of points into a number of clusters according to the underlying subspace structure is called *subspace clustering* (Elhamifar and Vidal, 2013). An extensive survey of subspace clustering methods can be found in Vidal (2011).

One type of subspace clustering methods that have gained much popularity in recent years is spectral-based methods (Liu et al., 2012; Lu et al., 2012; Elhamifar and Vidal, 2013; Li and Vidal, 2015; Huang et al., 2015). Spectral-based subspace clustering methods have been shown to enjoy excellent performance in numerous real world problems, including motion segmentation and face clustering. This type of methods solve for the cluster labels through a two-stage procedure. An affinity matrix is constructed in the first stage, and spectral clustering (Von Luxburg, 2007) is applied to the affinity matrix to obtain the cluster labels in the second stage. The main difference of various spectral-based methods lie in how the affinity matrix is constructed, but all of them use the *self-expressiveness model* (Li and Vidal, 2015). It is based on the premise that each point can be well represented as a linear combination of other points. As a result, the coefficients are then used to form the affinity matrix.

Sparse Simplex Representation (SSR) (Huang et al., 2013, 2015) is one such self-expressiveness model that minimises both the reconstruction error of the linear combination and the ℓ_1-norm of the coefficient vector. It imposes a simplex constraint on the coefficient vector, which means that all coefficient values are non-negative and they sum up to one. As a result of the simplex constraint, the ℓ_1-penalty term in the objective becomes constant, therefore does not play a role in the problem formulation.

To resolve this, we propose a modified version of SSR, named *Weighted Sparse Simplex Representation (WSSR)*, which introduces a weight matrix that encodes local neighbourhood information. It also means that the penalty term is no longer a constant, therefore it does exert an influence on the solution vector. The WSSR problem can be expressed as a standard constrained quadratic programming problem, and we propose two different approaches to solve it.

In practice, it is often time-consuming and expensive to validate the cluster performance if there exists a large number of unlabelled points. A feasible alternative for evaluating and thus improving the model performance is to obtain labels for a small amount of data. The process of clustering the data in the presence of a small amount of constraint information is called *constrained clustering* (Basu et al., 2008). We extend the WSSR problem formulation into a constrained clustering framework termed WSSR+, which involves a flexible weighting scheme that allows the WSSR problem formulation to incorporate the available labelling information.

Constrained clustering provides performance improvement on the labelled points, and potentially leads to better partitioning of the unlabelled points. However, it is a passive way of clustering in the sense that the labelling information is fixed and given a priori. As such, it is not guaranteed that the points that we have constraint information about are the most informative for clustering. In order to improve the cluster performance more effectively and efficiently with a limited amount of labelling information, we can resort to *active learning* (Settles, 2009).

Active learning explores the structures of the current clusters and queries informative data labels that would lead to effective performance improvement. Existing active learning methods mainly treat the development of active strategies and that of clustering methods separately (Wang and Davidson, 2010a; Lipor and Balzano, 2017; Xiong et al., 2017). In this work, we integrate these two components into one unified active learning and constrained clustering framework. To summarise, we make the following contributions:

- *WSSR.* We propose a weighted extension of the Sparse Simplex Representation problem. We show that it performs favourably against state-of-the-art spectral-based subspace clustering methods on both synthetic and real data.
- *WSSR+.* We extend the WSSR problem formulation to incorporate available labelling information, thus provide a flexible constrained clustering framework.
- *A unified active learning and constrained clustering framework.* We build a unified optimisation framework for subspace clustering that iteratively improves cluster performance through constrained clustering and queries the labels for informative points that would lead to effective performance improvement.

The rest of this chapter is organised as follows. In Section 4.2, we discuss some of the existing work in the literature in the areas of subspace clustering and constrained clustering. In Section 4.3 and 4.4, we propose the problem formulation of the Weighted Sparse Simplex Representation (WSSR) and discuss its theoretical properties. We discuss two approaches that can solve the problem, and present an integrated active learning and constrained clustering framework. We demonstrate the effectiveness of our proposed methodology on synthetic and real data in Section 4.6 and 4.7, and provide concluding remarks in Section 4.8.

4.2 Literature Review

This work is inspired by and built upon previous work in the fields of both *subspace clustering* and *constrained clustering*. There are generally four types of subspace clustering methods: iterative, spectral, algebraic, and statistical (Li et al., 2017). In particular, the literature that we review in this section are within the realm of spectral-based methods.

4.2.1 Subspace Clustering

We have previously provided motivation for why we will be focusing on spectral-based methods in Section 4.1. Spectral-based methods involve constructing the affinity matrix and applying spectral clustering to the affinity matrix to obtain the cluster labels. The affinity matrix is built using the self-expressiveness model. Given a data set $\mathcal{X} = \left\{\boldsymbol{x}_i \in \mathbb{R}^P\right\}_{i=1}^N$, the main premise of this model is that every point $\boldsymbol{x} \in \mathcal{X}$ can be well approximated by a linear combination of a few other points from the same subspace. Concretely, the self-expressiveness model can be expressed as the following optimisation problem for each point:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \|\boldsymbol{\varepsilon}\|_\kappa + \rho\,\|\boldsymbol{\beta}\|_l \\ \text{s.t.} \quad & \boldsymbol{x} = Y_{-i}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \end{aligned} \tag{4.2.1}$$

where $Y_{-i} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_{i+1}, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{P\times(N-1)}$, i.e. the data matrix without the i-th column corresponding to $\boldsymbol{x}_i$ to prevent the trivial solution of self-representation. For the rest of this chapter, we simplify the notation of Y_{-i} to Y when there is no ambiguity. Here $\boldsymbol{\beta}$ denotes the coefficient vector of the linear combination in representing $\boldsymbol{x}$; and $\boldsymbol{\varepsilon}$ represents the difference between $\boldsymbol{x}$ and the linear combination $Y\boldsymbol{\beta}$, which is the *reconstruction error* term. Different methods used different norms $\|\cdot\|_\kappa$ and $\|\cdot\|_l$ on the coefficient vector and the reconstruction error, and ρ is a penalty parameter on the coefficient vector.

Combining the coefficient vectors for all N points together, we obtain the *coefficient matrix* B as follows

$$B = \begin{bmatrix} 0 & \beta_{12} & \beta_{13} & \dots & \beta_{1N} \\ \beta_{21} & 0 & \beta_{23} & \dots & \beta_{2N} \\ \beta_{31} & \beta_{32} & 0 & \dots & \beta_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta_{N1} & \beta_{N2} & \beta_{N3} & \dots & 0 \end{bmatrix}, \tag{4.2.2}$$

where β_{ij} in B denotes the coefficient value in front of $\boldsymbol{x}_i$ in the linear combination that approximates $\boldsymbol{x}_j$. That is, the coefficient vectors are stored in the columns of B.

The formulation in Eq. (4.2.1) can be transformed into matrix form as follows

$$\begin{aligned} \min_{B} \quad & \|E\|_\kappa + \rho \|B\|_l \\ \text{s.t.} \quad & X = XB + E, \\ & \text{diag}(B) = \mathbf{0}, \end{aligned} \tag{4.2.3}$$

where $X \in \mathbb{R}^{P \times N}$ is the full data matrix, diag$(B) \in \mathbb{R}^N$ denotes the diagonal entries of B, and $\mathbf{0} \in \mathbb{R}^N$ is a vector of all zeros. After a representation matrix B is obtained, a common way to construct a non-negative symmetric affinity matrix A is through $A = \left(|B| + |B|^\intercal\right)/2$ (Huang et al., 2015; Li et al., 2017, 2018a). The final clustering labels can then be obtained by applying a standard spectral clustering algorithm (Shi and Malik, 2000; Ng et al., 2002) to the affinity matrix.

4.2.1.1 Sparse Subspace Clustering (SSC)

The difference of various spectral-based methods mainly lie in the choice of the regularisation on the reconstruction term $\|\cdot\|_\kappa$ and the coefficient values $\|\cdot\|_l$. For example, in **Sparse Subspace Clustering (SSC)** (Elhamifar and Vidal, 2013), the data matrix X is decomposed into three parts: the linear combination XB, the noise E, and an additional sparse outlying term Z. The ℓ_1-norm is applied to all the coefficient vectors in the columns of B and the sparse outlying entries in Z to encourage sparseness. The Frobenius norm is used on the noise matrix E, which is equivalent to applying the ℓ_2-norm to each of the columns of E. Concretely, the full SSC problem that takes into account both noise and sparse outlying entries can be expressed as follows

$$\begin{aligned} \min_{B} \quad & \|B\|_1 + \frac{\rho_e}{2}\|E\|_F^2 + \rho_z \|Z\|_1 \\ \text{s.t.} \quad & X = XB + E + Z, \\ & \operatorname{diag}(B) = \mathbf{0}. \end{aligned} \tag{4.2.4}$$

The noise-free vector-based representation of the SSC problem can be expressed as follows

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \|\boldsymbol{\beta}\|_1 \\ \text{s.t.} \quad & \boldsymbol{x} = Y\boldsymbol{\beta}. \end{aligned} \tag{4.2.5}$$

There exists a solution vector $\boldsymbol{\beta}$ that is sparse, and its non-zero entries correspond to the data points that are from the same subspace as $\boldsymbol{x}$. Such a solution is referred to as a *sparse subspace representation*. It has been shown in Elhamifar and Vidal (2013) that, in the absence of noise and outlying entries, SSC can successfully recover sparse subspace representations of data points that lie in a union of linear subspaces.

For independent subspaces, solving Eq. (4.2.5) always recovers the sparse subspace representations of the data without any assumption on the data distribution within each subspace, other than that the rank of the data within each subspace is known. For disjoint subspaces, it can be shown that the the ℓ_1-minimisation problem on B recovers the sparse subspace representations of the data under mild conditions.

4.2.1.2 Affine Sparse Subspace Clustering (ASSC)

Affine Sparse Subspace Clustering (ASSC) (Li et al., 2018a) is an adaptation of SSC that models data as a union of affine subspaces instead of linear subspaces. Any point $\boldsymbol{x}$ in an affine subspace of dimension P can be written as an affine combination of $(P + 1)$ other points from the same subspace. To deal with affine subspaces, ASSC includes an affine constraint which requires that the coefficient values in $\boldsymbol{\beta}$ sum up to one for every point $\boldsymbol{x} \in \mathcal{X}$. It models the self-expressiveness problem using the vector-based formulation in Eq. (4.2.1).

Similar to the vector-based formulation for SSC, the ℓ_1-norm is applied to the coefficient vector $\boldsymbol{\beta}$ and the ℓ_2-norm is applied to the noise vector $\boldsymbol{\varepsilon}$. Unlike SSC, ASSC does not include a separate term for sparse outlying entries. The ASSC problem (see Eq. (21) in Li et al. (2018a)) solves the following optimisation problem

$$\begin{aligned} \min_{\boldsymbol{\beta}_i} \quad & \|\boldsymbol{\beta}_i\|_1 + \frac{\rho}{2}\|\boldsymbol{\varepsilon}_i\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{x}_i = Y\boldsymbol{\beta}_i + \boldsymbol{\varepsilon}_i, \\ & \mathbf{1}^\mathsf{T}\boldsymbol{\beta}_i = 1, \end{aligned} \tag{4.2.6}$$

for $i \in \{1, \dots, N\}$.

4.2.1.3 Structured Sparse Subspace Clustering (S3C)

Structured Sparse Subspace Clustering (S3C) (Li and Vidal, 2015) is a unified optimisation framework that solves for the coefficient matrix B and the data partitioning simultaneously. It is an iterative procedure that alternates between feeding information about the current data partitioning into the self-expressiveness model, and using the coefficient matrix B obtained from the model to obtain a better partitioning of the data.

The data partitioning is represented using a *segmentation matrix* $Q \in \mathbb{R}^{N\times K}$ that contains a list of segmentation vectors $Q = [\boldsymbol{q}_1, \dots, \boldsymbol{q}_N]^\mathsf{T}$, where $Q_{ik} = 1$ if point $\boldsymbol{x}_i$ is assigned to cluster k ($k \in \{1, \dots, K\}$), and 0 otherwise. Based on the matrix-based self-expressiveness formulation in (4.2.3), S3C uses a subspace structured ℓ_1-norm $\|\cdot\|_{1,Q}$ on the coefficient matrix B defined as follows

$$\begin{aligned} \|B\|_{1,Q} &= \|B\|_1 + \alpha\|B\|_Q \\ &= \|B\|_1 + \alpha\|\Theta \odot B\|_1 \\ &= \sum_{i,j} |B_{ij}| \left(1 + \frac{\alpha}{2}\|\boldsymbol{q}_i - \boldsymbol{q}_j\|_2^2\right), \end{aligned} \tag{4.2.7}$$

in which $\Theta \in \mathbb{R}^{N\times N}$ describes the agreement between pairwise cluster labels, and we have $\Theta_{ij} = \frac{1}{2}\|\boldsymbol{q}_i - \boldsymbol{q}_j\|_2^2$. The operator $\odot$ is the Hadamard product (i.e. the element-wise product), and α is a user-defined trade-off parameter.

The above regularisation term on B incorporates information on the current data segmentation Q into the optimisation framework. It can be viewed as an ℓ_1-norm on the coefficient matrix B with an additional penalty when two points are assigned to different clusters according to the current segmentation matrix Q. The regularisation term on the reconstruction error $\|\cdot\|_\kappa$ is chosen by the user, depending on the prior knowledge about the noise pattern. The optimisation problem for S3C is expressed as follows:

$$\begin{aligned} \min_{B,E,Q} \quad & \|B\|_{1,Q} + \rho \|E\|_\kappa \\ \text{s.t.} \quad & X = XB + E, \\ & \text{diag}(B) = \mathbf{0}, \\ & Q \in \mathcal{Q}, \end{aligned} \tag{4.2.8}$$

in which $\mathcal{Q} = \left\{Q \in \{0,1\}^{N \times K} : Q\mathbf{1} = \mathbf{1} \text{ and rank}\,(Q) = K\right\}$.

The problem in Eq. (4.2.8) is solved iteratively by alternating between solving two sub-problems. The first sub-problem is to solve for B and E given the segmentation matrix Q. This sub-problem can be solved using the Alternating Direction Method of Multipliers (ADMM) (Boyd et al., 2011). The second sub-problem is to solve for the segmentation matrix Q given B and E. This sub-problem is solved by applying spectral clustering to the data affinity matrix $A = \left(|B| + |B|^\mathsf{T}\right)/2$. This iterative process stops either when a maximum iteration number is reached, or when the relative changes in Θ or B in two consecutive iterations is small enough.

4.2.1.4 Sparse Subspace Clustering by Orthogonal Matching Pursuit (SSC-OMP)

Another well-known spectral-based method that exhibits excellent performance in practice is a combination of SSC and Orthogonal Matching Pursuit (OMP) (Pati et al., 1993), named **SSC-OMP** (You et al., 2016). It has been shown to be both effective and efficient on digit recognition and face clustering applications. Recall that in the SSC objective, the ℓ_1-norm is applied to the columns of the coefficient matrix B and the ℓ_2-norm to the columns of the error matrix E. It can be computationally prohibitive to solve N self-expressiveness problems with $(N-1)$ variables.

To resolve this problem, a slightly different self-expressiveness model is used in SSC-OMP for each point $\boldsymbol{x} \in \mathcal{X}$:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \|\boldsymbol{\varepsilon}\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{x} = Y\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \\ & \|\boldsymbol{\beta}\|_0 \leqslant n. \end{aligned} \tag{4.2.9}$$

Unlike in SSC, SSC-OMP applies an ℓ_0-norm on the coefficient vector $\boldsymbol{\beta}$. It restricts the maximum number of non-zero values in $\boldsymbol{\beta}$ by a non-negative integer n. It can be shown that the above problem can be solved exactly using the OMP algorithm under certain conditions (Tropp, 2004; Davenport and Wakin, 2010).

OMP solves the problem greedily by selecting one column in Y at a time, and computes the coefficients for the selected column until n columns are selected. As such, it simultaneously chooses a subset of points for the sparse linear combination and solves for the coefficients for the chosen points. Although it is not obvious how n should be chosen, SSC-OMP is much faster than SSC and has competitive performance against state-of-the-art methods.

4.2.1.5 Least Squares Regression (LSR)

There are other methods that also base their formulations on the self-expressiveness model, but do not necessarily encourage sparsity in the solution. For example, in **Least Squares Regression (LSR)** (Lu et al., 2012), the Frobenius norm is used on both the coefficient matrix B and the reconstruction error E. This allows the problem to be solved analytically and efficiently. The problem formulation of LSR with noise is stated as follows:

$$\begin{aligned} \min_{B} \quad & \rho \|B\|_F^2 + \|E\|_F^2 \\ \text{s.t.} \quad & X = XB + E, \\ & \text{diag}(B) = \mathbf{0}. \end{aligned} \tag{4.2.10}$$

It is worth noting that if the constraint that requires the diagonal entries of B to be zeros is removed, then the problem becomes that of ridge regression (Hoerl and Kennard, 1970) which also has an analytical solution.

4.2.1.6 Smooth Representation Clustering (SMR)

Similarly, **Smooth Representation Clustering (SMR)** (Hu et al., 2014) is another spectral-based method that also uses the Frobenius norm on the reconstruction error E. Previously in LSR, the Frobenius norm is used on the coefficient matrix B to encourage a grouping effect among points from the same subspace. SMR enforces this grouping effect explicitly by incorporating the prior information on the pairwise similarities between points into the regularisation on the coefficient matrix B. The similarity information can be encapsulated using the graph Laplacian $L = D - W$, where $W \in \mathbb{R}^{N \times N}$ is the pairwise similarity matrix and D is the diagonal degree matrix (Von Luxburg, 2007). The k nearest neighbours (k-NN) graph is used in Hu et al. (2014) when constructing the similarity matrix, which has been shown to perform well in the experiments. To avoid numerical instability, L is enforced to be strictly positive definite through $\tilde{L} = L + \varepsilon I$, where ε is a small constant.

The matrix-based formulation of SMR can be expressed as follows

$$\begin{aligned} \min_{B} \quad & \rho \|E\|_F^2 + \text{tr}\left(B\tilde{L}B^{\mathsf{T}}\right) \\ \text{s.t.} \quad & X = XB + E, \\ & \text{diag}(B) = \mathbf{0}. \end{aligned} \tag{4.2.11}$$

It is worth noting that the term involving B with the trace norm resembles that of the objective in the normalised cut problem (See Eq. (2.3.16) in Section 2.3.3), whose solution can be approximated via spectral clustering. In the normalised cut objective, the matrix that we solve for is an approximation of the segmentation matrix.

4.2.2 Constrained Clustering

Sometimes there is a limited amount of external information available to potentially help improve the cluster performance. The external information is often given either in the form of 'must-link' and 'cannot-link' constraints, or in the form of class labels. Clustering with a limited amount of external information is called constrained clustering (Basu et al., 2008). There exists a vast amount of work in constrained clustering (Chapelle et al., 2006), and in particular, spectral-based constrained clustering (Kamvar et al., 2003; Li et al., 2009; Wang and Davidson, 2010b; Liu et al., 2018).

4.2.2.1 Spectral Learning

An early constrained spectral clustering algorithm developed by Kamvar et al. (2003) is called **Spectral Learning (SL)**. It has been successfully applied to the clustering of text data. Due to its motivation from document clustering, this model is called the "interested reader" model. The goal of document clustering is to identify the main topics in a set of documents. The main idea of this model is that, after an interested reader finishes reading one interesting article, he or she would move on to the next article that is highly similar to the article that has just been read.

The affinity matrix A contains the prior information about the pairwise similarities between articles. When there exists constraint information, it is proposed to modify the affinity matrix A such that $A_{ij} = A_{ji} = 1$ if two articles $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are from the same topic, and $A_{ij} = A_{ji} = 0$ otherwise. Given the affinity matrix A, it is proposed to transform the similarities into transition probabilities. That is, the probabilities of transitioning from one article to all remaining articles sum up to 1. The transition probabilities define a Markov chain (Gagniuc, 2017). It is proposed to calculate the

Markov transition matrix M as follows

$$M = \frac{1}{a_{\max}} \left(A + a_{\max} I - D \right), \tag{4.2.12}$$

where $a_{\max}$ is the maximum row sum of A, and D is the degree matrix of A. To see how this normalisation step transforms the entries in A into transition probabilities, one can show that the column sum for an arbitrary row i ($i \in \{1, \ldots, N\}$) in M is equal to one. This can be shown in the following:

$$\sum_{j=1}^{N} M_{ij} = \frac{1}{a_{\max}} \left(\sum_{j \neq i} A_{ij} - D_{ii} + a_{\max} \right) = \frac{a_{\max}}{a_{\max}} = 1, \quad \forall\, i \in \{1, \ldots, N\}, \tag{4.2.13}$$

in which the i-th diagonal entry in D is equal to the sum of all edge weights attached to $\boldsymbol{x}_i$ (i.e. $\sum_{j \neq i} A_{ij} = D_{ii}$) by definition. SL applies the normalised cut spectral clustering algorithm (Ng et al., 2002) to the K eigenvectors of M corresponding to its K largest eigenvalues.

4.2.2.2 Constrained Clustering via Spectral Regularisation (CCSR)

Unlike SL, which encodes constraint information by modifying the affinity matrix, Constrained Clustering via Spectral Regularisation (CCSR) (Li et al., 2009) incorporates pairwise constraint information by finding a better spectral embedding matrix $V \in \mathbb{R}^{N \times K}$. The spectral embedding matrix V is composed of the K eigenvectors $V = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_K]$, in which $\boldsymbol{v}_j$ is the j-th column in V that corresponds to the j-th smallest eigenvalue of the graph Laplacian matrix. Ideally, the spectral embedding matrix V should be as close to the data segmentation matrix Q as possible. In an ideal scenario where the spectral embedding matrix is the data segmentation matrix, the cluster labels can be trivially obtained through K-means clustering.

Given the constraint information that is available, the goal of CCSR is to obtain a better spectral embedding matrix than V (Li et al., 2009). Let $M = [\boldsymbol{m}_1, \ldots, \boldsymbol{m}_N]^{\mathsf{T}}$ be the new spectral embedding matrix, in which $\boldsymbol{m}_i$ represents the new spectral embedding vector for point $\boldsymbol{x}_i$. The new spectral embedding matrix M is of size N by d, in which d

is the desired dimension for representing the data. It is a user-defined parameter, which has been set to 15 in all experiments in Li et al. (2009).

We first discuss how the quality of a spectral embedding matrix M is measured. The following cost function is the proposed measure of agreement between M and the constraint information

$$L(M) = \sum_{i=1}^{N} \left(\boldsymbol{m}_i^{\mathsf{T}}\boldsymbol{m}_i - 1\right)^2 + \sum_{(i,j)\in\mathcal{S}_M} \left(\boldsymbol{m}_i^{\mathsf{T}}\boldsymbol{m}_j - 1\right)^2 + \sum_{(i,j)\in\mathcal{S}_C} \left(\boldsymbol{m}_i^{\mathsf{T}}\boldsymbol{m}_j - 0\right)^2, \tag{4.2.14}$$

in which $\mathcal{S}_M$ denotes the set of 'must-link' constraints, and $\mathcal{S}_C$ denotes the set of 'cannot-link' constraints. Let $\mathcal{S} = \{(i, j, c_{ij})\}$ denote the set of all pairwise constraints, where $c_{ij} = 1$ corresponds to a 'must-link' constraint and $c_{ij} = 0$ corresponds to a 'cannot-link' constraint. It is assumed that $(i, i, c_{ii}) \in \mathcal{S}$ where c_{ii} for $i \in \{1, \ldots, N\}$. As such, the cost function in Eq. (4.2.14) can be rewritten in a succinct manner as follows

$$L(M) = \sum_{(i,j,c_{ij})\in\mathcal{S}} \left(\boldsymbol{m}_i^{\mathsf{T}}\boldsymbol{m}_j - c_{ij}\right)^2. \tag{4.2.15}$$

We know that the eigenvectors in the columns of V are orthonormal to each other, thus constitute a basis for the spectral embedding space. It can be expressed as $\mathcal{H} = \left\{V_d F | F \in \mathbb{R}^{K\times d}\right\}$, in which V_d denotes the first d columns of the spectral embedding matrix V. The problem is then to find the $\boldsymbol{m}_i$ that minimises Eq. (4.2.15) within the spectral embedding space. Let $\boldsymbol{r}_i \in \mathbb{R}^K$ denote the i-th row in the spectral embedding matrix V, and we have $V = [\boldsymbol{r}_1, \ldots, \boldsymbol{r}_N]^{\mathsf{T}}$. Thus each row $\boldsymbol{m}_i$ in the modified spectral embedding matrix M can be obtained through $\boldsymbol{m}_i = F^{\mathsf{T}}\boldsymbol{r}_i$. As such, Eq. (4.2.15) can be rewritten as

$$L(F) = \sum_{(i,j,c_{ij})\in\mathcal{S}} \left(\boldsymbol{r}_i^{\mathsf{T}} F F^{\mathsf{T}} \boldsymbol{r}_j - c_{ij}\right)^2. \tag{4.2.16}$$

Thus, the problem of searching for a better spectral embedding matrix is transformed into the above problem of searching for F.

A relaxation of the above unconstrained minimisation problem can be solved via a semi-definite programme. Once F is obtained, we can obtain the modified spectral embedding matrix $M = VF$ and apply K-means clustering to M to obtain the final cluster labels.

4.2.2.3 Constrained Spectral Partitioning (CSP)

Another spectral-based algorithm **Constrained Spectral Partitioning (CSP)** (Wang and Davidson, 2010b; Wang et al., 2014) solves a modified version of the normalised cut spectral clustering problem (Shi and Malik, 2000; Von Luxburg, 2007). It introduces a pairwise constraint matrix $\Phi \in \mathbb{R}^{N\times N}$ in the problem formulation. Given the set of 'must-link' constraints $\mathcal{S}_M$ and the set of 'cannot-link' constraints $\mathcal{S}_C$, Φ is defined as:

$$\Phi_{ij} = \Phi_{ji} = \begin{cases} 1, & (i,j) \in \mathcal{S}_M, \\ -1, & (i,j) \in \mathcal{S}_C, \\ 0, & \text{no constraint information available.} \end{cases} \tag{4.2.17}$$

When $K = 2$, CSP solves the following optimisation problem:

$$\begin{aligned} \underset{\boldsymbol{g}\in\mathbb{R}^N}{\arg\min} \quad & \boldsymbol{g}^{\mathsf{T}} L_{\text{sym}} \boldsymbol{g} \\ \text{s.t.} \quad & \boldsymbol{g}^{\mathsf{T}} \Phi_{\text{sym}} \boldsymbol{g} \geq \alpha, \\ & \boldsymbol{g}^{\mathsf{T}} \boldsymbol{g} = \text{vol}, \\ & \boldsymbol{g} \neq D^{\frac{1}{2}} \mathbf{1}, \end{aligned} \tag{4.2.18}$$

where α is a threshold for how well the user would like the constraints to be satisfied. Here $L_{\text{sym}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ is the normalised symmetric graph Laplacian matrix, $\Phi_{\text{sym}} = D^{\frac{1}{2}} \Phi D^{\frac{1}{2}}$ is the normalised constraint matrix, and $\text{vol} = \sum_{i=1}^{N} D_{ii}$ is the sum of the diagonal entries in the degree matrix.

The cluster labels can be easily determined by applying K-means clustering to the solution vector to the generalised eigenvector problem in Eq. (4.2.18). It can be shown that the cluster labels for $K > 2$ can be obtained by applying K-means clustering to the top K generalised eigenvectors. When there is no constraint information available, the CSP problem reduces to the normalised cut spectral clustering algorithm in Shi and

Malik (2000).

4.2.2.4 Partition Level Constrained Clustering (PLCC)

Recently developed constrained clustering framework **Partition Level Constrained Clustering (PLCC)** (Liu and Fu, 2015; Liu et al., 2018) is applicable to both K-means clustering and spectral clustering. PLCC incorporates available constraints through a side information matrix $S \in \mathbb{R}^{N \times K}$, where $S_{ik} = 1$ if point $\boldsymbol{x}_i$ belongs to cluster k and zero otherwise. After incorporating the constraint information, the spectral embedding of the data can be obtained through eigen-decomposition of the following

$$D^{\frac{1}{2}} W D^{\frac{1}{2}} + \lambda S S^{\mathsf{T}}, \tag{4.2.19}$$

in which W is the pairwise similarity matrix, and λ is a penalty term that indicates how much belief we have on the side information matrix. In the experiments in Liu et al. (2018), λ is set to be 100. The cluster labels can be obtained by applying K-means clustering to the K eigenvectors that correspond to the K largest eigenvalues of Eq. (4.2.19).

4.2.2.5 Constrained Structured Sparse Subspace Clustering (CS3C)

All of the aforementioned methods concern the general problem of constrained spectral clustering. Recent research has been carried out on the combination of subspace clustering and constrained clustering specifically. **Constrained Structured Sparse Subspace Clustering (CS3C)** (Li et al., 2017) is an adaptation of S3C, as introduced in Section 4.2.1.3, that modifies the subspace structured ℓ_1-norm $\|B\|_{1,Q}$ on the coefficient matrix B to incorporate the side information as follows:

$$\begin{aligned} \|B\|_{\Psi,Q} &= \|B \odot \Psi\|_1 + \alpha \|B\|_Q \\ &= \sum_{ij} |B_{ij}| \left(\Psi_{ij} + \frac{1}{2} \|\boldsymbol{q}_i - \boldsymbol{q}_j\|_2^2 \right), \end{aligned} \tag{4.2.20}$$

where Ψ is the side information matrix defined as follows:

$$\Psi_{ij} = \Psi_{ji} = \begin{cases} \exp(-1), & (i,j) \in \mathcal{S}_M, \\ \exp(0), & \text{no constraint information available,} \\ \exp(1), & (i,j) \in \mathcal{S}_C. \end{cases} \tag{4.2.21}$$

A further development in Li et al. (2018b), termed CS3C+, combines the stage of obtaining the coefficient matrix with the spectral clustering stage. Apart from solving the CS3C problem to obtain the coefficient matrix, CS3C+ also satisfies the constraints in the spectral clustering stage by applying constrained K-means clustering (Wagstaff et al., 2001) to the spectral embedding of the data.

4.3 Weighted Sparse Simplex Representation (WSSR)

In this section, we first provide a detailed introduction to Sparse Simplex Representation (SSR) (Huang et al., 2015) in Section 4.3.1. We discuss some of the drawbacks of SSR, and propose a modified version in Section 4.3.2 that addresses the drawbacks, named Weighted Sparse Simplex Representation (WSSR). In Section 4.5, we present an extension of WSSR, named WSSR+, that lends itself naturally in a constrained clustering and active learning framework.

4.3.1 Sparse Simplex Representation (SSR)

The sparse simplex representation (SSR) model is first proposed in Huang et al. (2013) for the purpose of modelling brain networks. It represents each data point as a convex combination of other data points. The requirement that the coefficients have to be non-negative allows for a probabilistic interpretation on the strength of links and can directly play the role of pairwise similarities. The Sparse Simplex Representation (SSR) model solves the following optimisation programme for each data point (Huang et al., 2015):

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \frac{1}{2} \left\| \boldsymbol{x} - Y\boldsymbol{\beta} \right\|_2^2 + \rho \left\| \boldsymbol{\beta} \right\|_1 \\ \text{s.t.} \quad & \left\| \boldsymbol{\beta} \right\|_1 = 1, \\ & \boldsymbol{\beta} \succeq \mathbf{0}, \end{aligned} \tag{4.3.1}$$

in which the last line states that every entry in $\boldsymbol{\beta}$ is greater than or equal to zero.

Given the objective function, the simplex constraint makes the coefficient vector shift invariant. As a result, the ℓ_1-penalty term on the coefficient vector does not influence the solution of the problem. It is proposed in Huang et al. (2013) to use the accelerated projected gradient method to solve the SSR problem in Eq. (4.3.1). It is a standard quadratic programming problem that is also known as the constrained lasso (Gaines et al., 2018).

4.3.2 Weighted Sparse Simplex Representation (WSSR)

In order to make the ℓ_1-penalty term promote sparsity in the optimisation programme, we introduce a diagonal weight matrix $\Gamma \in \mathbb{R}^{(N-1)\times(N-1)}$ within the ℓ_1-term. The i-th ($i \in \{1, \dots, (N-1)\}$) diagonal entry in Γ, Γ_{ii}, denotes the pairwise dissimilarity between $\boldsymbol{x}$ and $\boldsymbol{x}_i$. The introduction of a weight matrix not only enables the ℓ_1-term to induce sparsity, but it also encourages more similar points to be favoured in the convex combination.

The idea of incorporating proximity information into the optimisation problem has been previously utilised in Elhamifar and Vidal (2011) in the setting of manifold clustering. The authors introduce a proximity inducing matrix in which the weights are calculated based on the pairwise Euclidean distance. There is a number of options available for calculating the weight matrix. Elhamifar and Vidal (2011) first normalise the points to unit length, and then calculate the weights based on the pairwise Euclidean distance between points and normalise the weights to be between zero and one.

We use the inverse absolute cosine similarity to measure the pairwise proximity between points, i.e. $\Gamma_{ii} = \frac{1}{|\boldsymbol{x}^\mathsf{T}\boldsymbol{x}_i|}$ ($\boldsymbol{x}_i \neq \boldsymbol{x}$). The rationale behind this can be illustrated with the synthetic data example in Figure 4.3.1, which shows a data set with three one-dimensional subspaces before and after the data points are normalised to unit length. Firstly, when the data points are normalised to lie on the unit sphere, the points that are from the same cluster are condensed into two opposite regions on the sphere. This ensures that the locations of the non-zero entries in $\boldsymbol{\beta}$ do not depend on whether the points are close to or far away from $\boldsymbol{x}$ (Elhamifar and Vidal, 2011; You et al., 2016). Secondly, it is clear that points that are from the same cluster have very small absolute cosine similarity to each other. Therefore, the inverse absolute cosine similarity is an appropriate measure for evaluating the dissimilarity between points across different clusters. We remove any point $\boldsymbol{x}_i$ that has zero cosine similarity with $\boldsymbol{x}$, so that points that are not similar to $\boldsymbol{x}$ at all are not considered.

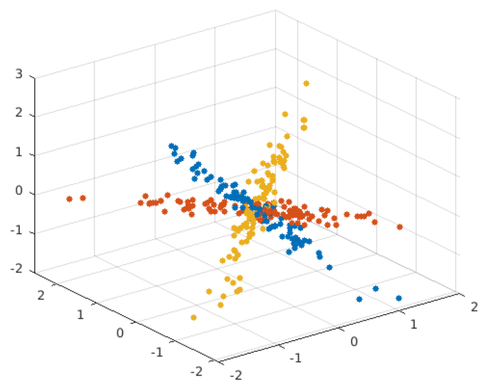
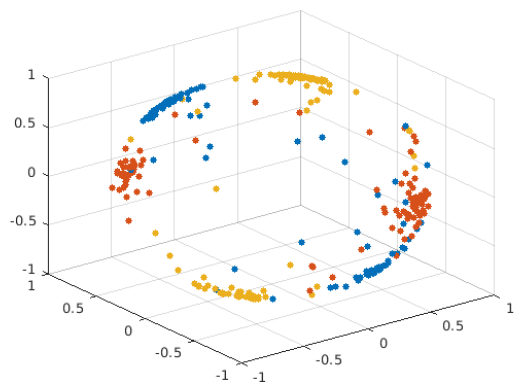



Figure 4.3.1: An illustration of the effect of the data normalisation step, which provides the rationale for the use of the inverse cosine similarity. **Left:** The original data points. **Right:** The data points that have been normalised to lie on the unit sphere.

Although the ℓ_1-term now plays an active role in the problem, there are several drawbacks with this formulation. Firstly, the ℓ_1-penalty term in (4.3.1) is not a strictly convex function and the solution is not always unique (Zou and Hastie, 2005). Secondly, the lasso chooses one point to have non-zero coefficient among a set of correlated points. This implies that if there exists a few correlated points (points that are from the same subspace as $\boldsymbol{x}$), only one of them might be selected to have non-zero coefficient. This is undesirable, as it would lead to an overly sparse coefficient matrix thus over-segmentation of the data. Ideally, we would like each point to be represented as a convex combination of a few points from the same subspace.

We can resolve these issues by adding an additional ℓ_2-penalty in the objective. The combination of the ℓ_1 and ℓ_2 penalty is called an elastic net penalty (Zou and Hastie, 2005). The elastic net penalty form in Eq. (4.3.2) makes the WSSR problem strictly convex, thus yielding a unique solution. It also encourages a grouping effect, which means that it either chooses groups of variables (in our case 'data points') together that are correlated or not at all. This property is especially useful in the clustering setting, as the purpose in using sparse regression is to choose those points that potentially come from the same cluster as the response variable $\boldsymbol{x}$ (Zou and Hastie, 2005; Segal et al., 2003). The ℓ_2-term in the objective helps to improve both computational efficiency and effectiveness, as has been noted previously in Tibshirani (2011). Concretely, we propose to solve the following problem for every point $\boldsymbol{x} \in \mathcal{X}$:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \frac{1}{2}\|\boldsymbol{x} - Y\boldsymbol{\beta}\|_2^2 + \frac{\varepsilon}{2}\|\Gamma\boldsymbol{\beta}\|_2^2 + \rho\|\Gamma\boldsymbol{\beta}\|_1 \\ \text{s.t.} \quad & \boldsymbol{\beta}^\mathsf{T}\mathbf{1} = 1, \\ & \boldsymbol{\beta} \succeq \mathbf{0}. \end{aligned} \tag{4.3.2}$$

We name this problem formulation the **Weighted Sparse Simplex Representation (WSSR)**. The problem formulation in Eq. (4.3.2) can be rewritten in the following quadratic programming form:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \frac{1}{2}\boldsymbol{\beta}^\mathsf{T}\left(Y^\mathsf{T}Y + \varepsilon\Gamma^\mathsf{T}\Gamma\right)\boldsymbol{\beta} + \left(\rho\boldsymbol{\gamma} - Y^\mathsf{T}\boldsymbol{x}\right)^\mathsf{T}\boldsymbol{\beta} \\ \text{s.t.} \quad & \boldsymbol{\beta}^\mathsf{T}\mathbf{1} = 1, \\ & \boldsymbol{\beta} \succeq \mathbf{0}, \end{aligned} \tag{4.3.3}$$

in which $\boldsymbol{\gamma} = \text{diag}(\Gamma) \in \mathbb{R}^{(N-1)}$ is called the weight vector, whose entries correspond to the diagonal values of Γ. The value of ε is generally chosen to be very small, e.g. 10^{-4} (Gaines et al., 2018). The problem can be solved by a standard constrained quadratic programming solver.

Stretching the columns in Y. There are two main goals that we want to achieve in the WSSR problem. One is to encourage sparsity of the solution vector $\boldsymbol{\beta}$, that is, to select a few columns in Y. Another is to reduce the reconstruction error, $\|\boldsymbol{x} - Y\boldsymbol{\beta}\|_2$. So far, the only preprocessing step we do is to normalise the data points that are not orthogonal to $\boldsymbol{x}$ to unit length. However, this normalisation step has certain implications on the WSSR problem.

Firstly, when choosing more than one point from the probability simplex into the convex combination, the reconstructed point is unavoidably shorter than the original point. This effect can be illustrated in Figure 4.3.2 below. As we can see, if we use the

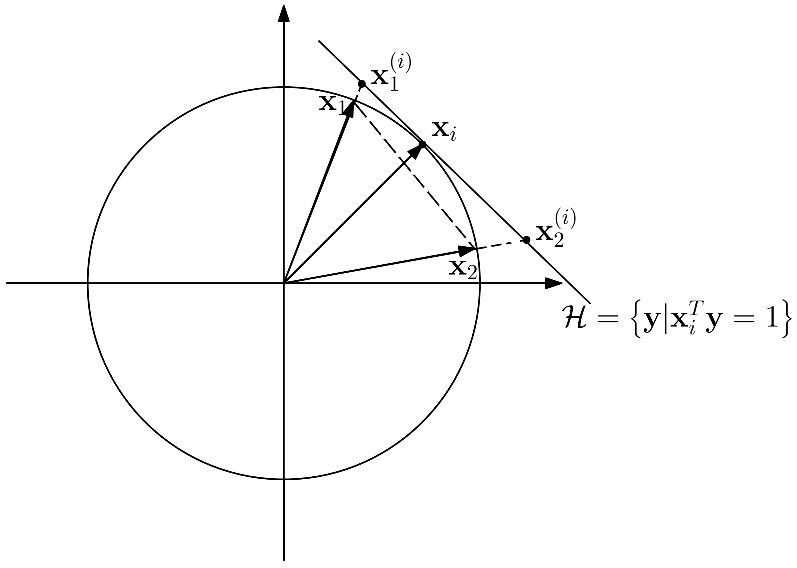


Figure 4.3.2: A geometric illustration of the necessity for stretching points in Y.

convex combination of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ to reconstruct $\boldsymbol{x}_i$ which lies on the unit sphere, then the potential reconstructed point can only touch the dotted line connecting $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ instead of on the unit sphere. Secondly, as we incrementally include more points (ordered by their dissimilarity to $\boldsymbol{x}_i$) into the convex combination, the length of the reconstructed vector is non-increasing. Since we want to pursue the goals of both inducing sparsity and reducing reconstruction error, we extend both $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ such that the potential reconstructed point lies anywhere on the perpendicular hyperplane of $\boldsymbol{x}_i$. Let us denote $\boldsymbol{x}_j^{(i)}$ as a point on the perpendicular hyperplane $\mathcal{H}$ of $\boldsymbol{x}_i$, which is obtained by stretching $\boldsymbol{x}_j$. We have that

$$\boldsymbol{x}_i^{\mathsf{T}} \left(\boldsymbol{x}_j^{(i)} - \boldsymbol{x}_i \right) = 0. \tag{4.3.4}$$

If we let $\boldsymbol{x}_j^{(i)} = t_j \boldsymbol{x}_j$ where t_j is the stretching constant for $\boldsymbol{x}_j$, then $t_j = \left(\boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{x}_j \right)^{-1}$.

Subspace Clustering by WSSR. Collating the above discussion, we summarise the whole procedure of subspace clustering using WSSR in Algorithm 7. In the last step of the algorithm, we adopt a common and simple approach to obtain the data affinity matrix A, which has been used in a few previously proposed subspace methods (Huang et al., 2015; Li et al., 2017, 2018a). There are several other ways of transforming the (not necessarily non-negative) coefficient matrix B into a non-negative symmetric data affinity matrix A. For example, we can simply add up $|B|$ and $|B|^\mathsf{T}$ as is done in (Elhamifar and Vidal, 2013), or we can explore the merit of grouping effect by multiplying the individual coefficient vectors as is done in (Hu et al., 2014).

Algorithm 7: Weighted Sparse Simplex Representation (WSSR)

```
Input :Set of data points: $\mathcal{X}$
       Number of clusters: $K$
       Neighbourhood size: $k$
For $\boldsymbol{x} \in \mathcal{X}$:
1. Compute the weight vector $\boldsymbol{\gamma} \in \mathbb{R}^k$ for the $k$ nearest neighbours of $\boldsymbol{x}$
2. Normalise and stretch each column vector in $Y$
3. Solve the WSSR problem in (4.3.3) to obtain the coefficient vector $\boldsymbol{\beta}$
End
- Combine all $\boldsymbol{\beta}$s to obtain the coefficient matrix $B \in \mathbb{R}^{N\times N}$ according to
  Eq. (4.2.2)
- Apply normalised cut spectral clustering (Ng et al., 2002) to the affinity matrix
  $A = \frac{1}{2}\left(|B| + |B|^\mathsf{T}\right)$ to obtain $K$ clusters
```

4.4 Properties of WSSR

In this section, we first study the optimality conditions for the WSSR problem in Section 4.4.1. It provides us with further information on the characteristics of the optimal solution vector. In Section 4.4.2, we present an analytical solution to a sub-problem of WSSR without the non-negativity constraint. In addition, we also propose a projected gradient descent approach to solve the full WSSR problem. In Section 4.4.3 and 4.4.4, we investigate the necessary and sufficient conditions under which the trivial solution is obtained. That is, only the most similar point is chosen and has coefficient one.

4.4.1 KKT Conditions for Optimality

One of the *constraint qualifications* one can use to guarantee strong duality is *Slater's condition* (Boyd and Vandenberghe, 2004). A problem satisfies Slater's condition if there exists a point that is strictly feasible in the optimisation programme (i.e. all constraints are satisfied and the inequality constraints are satisfied with strict inequalities). In the WSSR problem as stated in Eq. (4.3.3), Slater's condition says that strong duality holds if there exists a $\boldsymbol{\beta}$ with $\boldsymbol{\beta} \succ \mathbf{0}$ and $\boldsymbol{\beta}^\mathsf{T}\mathbf{1} = 1$. This is indeed satisfied, which can be shown with a trivial example in which $\boldsymbol{\beta} = \left[\frac{1}{N-1}, \ldots, \frac{1}{N-1}\right]$ where $\boldsymbol{\beta} \in \mathbb{R}^{(N-1)}$. For any optimisation problem with differentiable objective and constraint functions for which strong duality holds, the Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient conditions for obtaining the optimal solution (Boyd and Vandenberghe, 2004).

Firstly, the stationarity condition in the KKT conditions states that when optimality is achieved, the derivative of the Lagrangian with respect to $\boldsymbol{\beta}$ is zero. The Lagrangian $L\left(\boldsymbol{\beta}; \lambda, \boldsymbol{\mu}\right)$ associated with the WSSR problem in Eq. (4.3.3) can be expressed as

$$L(\boldsymbol{\beta}; \lambda, \boldsymbol{\mu}) = \frac{1}{2}\boldsymbol{\beta}^\mathsf{T}\left(Y^\mathsf{T}Y + \varepsilon\Gamma^\mathsf{T}\Gamma\right)\boldsymbol{\beta} + \left(\rho\boldsymbol{\gamma} - Y^\mathsf{T}\boldsymbol{x}\right)^\mathsf{T}\boldsymbol{\beta} - \boldsymbol{\mu}^\mathsf{T}\boldsymbol{\beta} + \lambda\left(\boldsymbol{\beta}^\mathsf{T}\mathbf{1} - 1\right), \tag{4.4.1}$$

in which λ is a scalar and $\boldsymbol{\mu}$ is a vector of non-negative Lagrange multipliers. Thus, the stationarity condition gives the following

$$\nabla L(\boldsymbol{\beta}; \lambda, \boldsymbol{\mu}) = \left(Y^\mathsf{T}Y + \varepsilon\Gamma^\mathsf{T}\Gamma\right)\boldsymbol{\beta} - Y^\mathsf{T}\boldsymbol{x} + \rho\boldsymbol{\gamma} + \lambda\mathbf{1} - \boldsymbol{\mu} = \mathbf{0}, \tag{4.4.2}$$

which can be simplified to

$$\boldsymbol{\beta} = \left(Y^\mathsf{T}Y + \varepsilon\Gamma^\mathsf{T}\Gamma\right)^{-1}\left(Y^\mathsf{T}\boldsymbol{x} + \boldsymbol{\mu} - \rho\boldsymbol{\gamma} + \lambda\mathbf{1}\right). \tag{4.4.3}$$

Since all diagonal entries in Γ are positive, we have that the matrix $\left(Y^\mathsf{T}Y + \varepsilon\Gamma^\mathsf{T}\Gamma\right)$ is full rank thus invertible. Secondly, the KKT conditions state that any primal optimal $\boldsymbol{\beta}$ must satisfy both the equality and inequality constraints in Eq. (4.3.3). In addition, any dual optimal λ and $\boldsymbol{\mu}$ must satisfy the dual feasibility constraint $\boldsymbol{\mu} \succeq \mathbf{0}$. Thirdly, the KKT conditions state that the following holds

$$\mu_i \beta_i = 0, \quad \forall\, i \in \{1, \ldots, (N-1)\},$$

for any primal optimal $\boldsymbol{\beta}$ and dual optimal $\boldsymbol{\mu}$ when strong duality holds. This is called the complementary slackness condition.

To put everything together, when strong duality holds, any primal optimal $\boldsymbol{\beta}$ and any dual optimal λ and $\boldsymbol{\mu}$ must satisfy the following KKT conditions:

Stationarity: $\boldsymbol{\beta} = \left(Y^\mathsf{T} Y + \varepsilon \Gamma^\mathsf{T} \Gamma\right)^{-1} \left(Y^\mathsf{T} \boldsymbol{x} + \boldsymbol{\mu} - \rho \boldsymbol{\gamma} + \lambda \mathbf{1}\right)$,

Equality constraint: $\boldsymbol{\beta}^\mathsf{T} \mathbf{1} = 1$,

Inequality constraint: $\boldsymbol{\beta} \succeq \mathbf{0}$,

Dual feasibility: $\boldsymbol{\mu} \succeq \mathbf{0}$,

Complementary slackness: $\mu_i \beta_i = 0 \quad \forall\, i \in \{1, \ldots, (N-1)\}$.

4.4.2 Solving the Full WSSR Problem

As a first step, we consider the sub-problem of WSSR without the non-negativity constraint:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \frac{1}{2} \boldsymbol{\beta}^\mathsf{T} \left(Y^\mathsf{T} Y + \varepsilon \Gamma^\mathsf{T} \Gamma\right) \boldsymbol{\beta} + \left(\rho \boldsymbol{\gamma} - Y^\mathsf{T} \boldsymbol{x}\right)^\mathsf{T} \boldsymbol{\beta} \\ \text{s.t.} \quad & \boldsymbol{\beta}^\mathsf{T} \mathbf{1} = 1. \end{aligned} \tag{4.4.4}$$

Setting the derivative of the Lagrangian for (4.4.4) to zero and satisfying the equality constraint give us the following linear system of equations:

$$\begin{bmatrix} Y^\mathsf{T} Y + \varepsilon \Gamma^\mathsf{T} \Gamma & \mathbf{1} \\ \mathbf{1}^\mathsf{T} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta} \\ \lambda \end{bmatrix} = \begin{bmatrix} Y^\mathsf{T} \boldsymbol{x} - \rho \boldsymbol{\gamma} \\ 1 \end{bmatrix}. \tag{4.4.5}$$

Solving the set of linear equations gives the optimal primal and dual variables for Eq. (4.4.4). It is worth noting that the solution vector of the sub-problem does not induce sparsity, and we obtain real values in the solution vectors. We refer to this version as the **Weighted Smooth Representation (WSR)**.

If we project the solution vector of Eq. (4.4.4) to the probability simplex, then both of the constraints in the full WSSR problem would be satisfied. However, the projected solution vector might not be the best one in the probability simplex in terms of minimising the original objective. In the rest of this section, we detail a projected gradient descent algorithm that solves the full problem iteratively.

We can cast the WSSR problem as an unconstrained minimisation problem

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{(N-1)}} f(\boldsymbol{\beta}) + \mathbb{1}_{\Delta^{(N-1)}}(\boldsymbol{\beta}), \tag{4.4.6}$$

where $f(\boldsymbol{\beta})$ is the WSSR objective as stated in Eq. (4.3.3), and $\mathbb{1}_{\Delta^{(N-1)}}(\boldsymbol{\beta})$ is the indicator function in which $\Delta^{(N-1)}$ denotes the $(N-1)$-dimensional probability simplex given by the constraints. We can express $\mathbb{1}_{\Delta^{(N-1)}}(\boldsymbol{\beta})$ as follows

$$\mathbb{1}_{\Delta^{(N-1)}}(\boldsymbol{\beta}) = \begin{cases} 0, & \boldsymbol{\beta}\in\Delta^{(N-1)}, \\ +\infty, & \text{otherwise.} \end{cases}$$

Problem Eq. (4.4.6) can be solved via the *proximal gradient method* (Parikh and Boyd, 2014) as

$$\boldsymbol{\beta}^{t+1} = \text{prox}_{\mathbb{1}_{\Delta^{(N-1)}}}\left(\boldsymbol{\beta}^t - \eta^t \nabla f(\boldsymbol{\beta}^t)\right), \tag{4.4.7}$$

where η^t is the step size for iteration t $(t = 1, 2, 3, \ldots)$. The derivative of the objective function at $\boldsymbol{\beta}^t$ is given by

$$\nabla f(\boldsymbol{\beta}^t) = \left(Y^\mathsf{T} Y + \varepsilon \Gamma^\mathsf{T}\Gamma\right)\boldsymbol{\beta}^t - Y^\mathsf{T}\boldsymbol{x} + \rho\boldsymbol{\gamma}. \tag{4.4.8}$$

The proximal operator in Eq. (4.4.7) reduces to the Euclidean projection onto the unit simplex

$$\boldsymbol{\beta}^{t+1} = \arg\min_{\boldsymbol{\beta}\in\Delta^{(N-1)}} \frac{1}{2}\left\|\boldsymbol{\beta} - \left(\boldsymbol{\beta}^t - \eta^t\nabla f(\boldsymbol{\beta}^t)\right)\right\|_2^2. \tag{4.4.9}$$

Projection onto the probability simplex can be achieved via a simple algorithm, see for example Chen and Ye (2011) and Wang and Carreira-Perpinán (2013). We adopt the projection algorithm proposed in Wang and Carreira-Perpinán (2013), which has a computational complexity of $\mathcal{O}\left(d \cdot \log\left(d\right)\right)$, with d being the dimension of the probability simplex which is upper bounded by $(N-1)$. In our case, d is equal to the length of the solution vector $\boldsymbol{\beta}$. One of the inputs we require is the step size for the gradient update step. There are many ways of determining the step size, one can either use a constant step size or diminishing step size. We use a diminishing step size that is square summable but not summable, $\eta^t = 1/(b+t)$, where b is a user-defined parameter. The algorithmic form of the projected gradient descent algorithm to find the coefficient vector $\boldsymbol{\beta}$ for any point $\boldsymbol{x} \in \mathcal{X}$ can be found in Algorithm 8.

Algorithm 8: Weighted Sparse Simplex Representation - Projected Gradient Descent (WSSR-PGD)

```
Input :Set of data points: $\mathcal{X}$
       Number of clusters: $K$
       Neighbourhood size: $k$
       Step size parameter: $b$
       Stopping criterion: $\delta$
Initialisation:
    • Obtain the analytical solution $\boldsymbol{\beta}^0$ to the sub-problem in Eq. (4.4.4)
    • Project $\boldsymbol{\beta}^0$ to the probability simplex to obtain $\boldsymbol{\beta}^1$
For $t = 1, 2, 3, \ldots$ repeat
    1. Gradient update step: $\boldsymbol{\phi}^{t+1} = \boldsymbol{\beta}^t - \eta^t \nabla f(\boldsymbol{\beta}^t)$, where
    $\nabla f(\boldsymbol{\beta}^t) = \left(Y^\top Y + \varepsilon \Gamma^\top \Gamma\right) \boldsymbol{\beta}^t - Y^\top \boldsymbol{x} + \rho \boldsymbol{\gamma}$
    2. Euclidean projection onto the probability simplex:
    $\boldsymbol{\beta}^{t+1} = \arg\min_{\boldsymbol{\beta} \in \Delta^{(N-1)}} \|\boldsymbol{\beta} - \boldsymbol{\phi}^{t+1}\|_2^2$
    3. Diminish the step size: $\eta^{t+1} = \frac{1}{b+t}$
until $f\left(\boldsymbol{\beta}^{t+1}\right) - f\left(\boldsymbol{\beta}^t\right) \leqslant \delta$
```

4.4.3 Necessary Condition for the Trivial Solution

Consider the WSSR problem formulation in Eq. (4.3.3) for a given $\boldsymbol{x} \in \mathcal{X}$, which we restate below:

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \frac{1}{2}\boldsymbol{\beta}^{\mathsf{T}}\left(Y^{\mathsf{T}}Y + \varepsilon\Gamma^{\mathsf{T}}\Gamma\right)\boldsymbol{\beta} + \left(\rho\boldsymbol{\gamma} - Y^{\mathsf{T}}\boldsymbol{x}\right)^{\mathsf{T}}\boldsymbol{\beta} \\ \text{s.t.} \quad & \boldsymbol{\beta}^{\mathsf{T}}\mathbf{1} = 1, \\ & \boldsymbol{\beta} \succeq \mathbf{0}. \end{aligned} \tag{4.4.10}$$

Without loss of generality, we assume that $Y = \left[\boldsymbol{x}_{(1)}, \boldsymbol{x}_{(2)}, \dots, \boldsymbol{x}_{(N-1)}\right]$ where $\boldsymbol{x}_{(k)}$ ($k \in \{1, 2, \dots, (N-1)\}$) is the k-th nearest neighbour of $\boldsymbol{x}$. Similarly $\boldsymbol{\gamma} = \text{diag}(\Gamma) = \left[\gamma_{(1)}, \gamma_{(2)}, \dots, \gamma_{(N-1)}\right]^{\mathsf{T}}$. Let $\boldsymbol{\beta}^{\star}$ denote the optimal solution to Eq. (4.4.10), we establish the necessary condition for the trivial solution $\|\boldsymbol{\beta}^{\star}\|_{\infty} = 1$ in Proposition 4.4.1.

Proposition 4.4.1. Assume the nearest neighbour of $\boldsymbol{x}$ is unique, i.e. $\boldsymbol{x}_{(1)} \neq \boldsymbol{x}_j$ for $j \neq (1)$. If the solution of the WSSR problem in Eq. (4.4.10) is given by $\boldsymbol{\beta}^{\star} = \boldsymbol{e}_1 = [1, 0, \dots, 0]^{\mathsf{T}} \in \mathbb{R}^{(N-1)}$, then the following holds

$$\rho > \max\left\{0, \max_{j \in \{2, \dots, (N-1)\}} \frac{(\boldsymbol{x}_{(1)} - \boldsymbol{x}_{(j)})^T(\boldsymbol{x}_{(1)} - \boldsymbol{x}) + \varepsilon\gamma_{(1)}^2}{\gamma_{(j)} - \gamma_{(1)}}\right\}. \tag{4.4.11}$$

Proof. To establish the above claim, it suffices to show that the directional derivative of the objective function at $\boldsymbol{e}_1$ is positive for all feasible directions in the unit simplex $\Delta^{(N-1)}$. We start by denoting the objective function value in Eq. (4.4.10) as $f(\boldsymbol{\beta})$. The derivative of the objective function is

$$\nabla f(\boldsymbol{\beta}) = (Y^{\mathsf{T}}Y + \varepsilon\Gamma^{\mathsf{T}}\Gamma)\boldsymbol{\beta} + \rho\boldsymbol{\gamma} - Y^{\mathsf{T}}\boldsymbol{x} = H\boldsymbol{\beta} + \rho\begin{bmatrix}\gamma_{(1)} \\ \gamma_{(2)} \\ \vdots \\ \gamma_{(N-1)}\end{bmatrix} - \begin{bmatrix}\boldsymbol{x}_{(1)}^{\mathsf{T}}\boldsymbol{x} \\ \boldsymbol{x}_{(2)}^{\mathsf{T}}\boldsymbol{x} \\ \vdots \\ \boldsymbol{x}_{(N-1)}^{\mathsf{T}}\boldsymbol{x}\end{bmatrix},$$

where

$$H = \begin{bmatrix} \boldsymbol{x}_{(1)}^{\mathsf{T}}\boldsymbol{x}_{(1)} + \varepsilon\gamma_{(1)}^2 & \boldsymbol{x}_{(1)}^{\mathsf{T}}\boldsymbol{x}_{(2)} & \cdots & \boldsymbol{x}_{(1)}^{\mathsf{T}}\boldsymbol{x}_{(N-1)} \\ \boldsymbol{x}_{(2)}^{\mathsf{T}}\boldsymbol{x}_{(1)} & \boldsymbol{x}_{(1)}^{\mathsf{T}}\boldsymbol{x}_{(2)} + \varepsilon\gamma_{(2)}^2 & \cdots & \boldsymbol{x}_{(2)}^{\mathsf{T}}\boldsymbol{x}_{(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{x}_{(N-1)}^{\mathsf{T}}\boldsymbol{x}_{(1)} & \boldsymbol{x}_{(N-1)}^{\mathsf{T}}\boldsymbol{x}_{(2)} & \cdots & \boldsymbol{x}_{(N-1)}^{\mathsf{T}}\boldsymbol{x}_{(N-1)} + \varepsilon\gamma_{(N-1)}^2 \end{bmatrix}.$$

Therefore $\nabla f(\boldsymbol{e}_1)$ is equal to

$$\nabla f(\boldsymbol{e}_1) = \begin{bmatrix} \boldsymbol{x}_{(1)}^{\mathsf{T}}\boldsymbol{x}_{(1)} + \varepsilon\gamma_{(1)}^2 \\ \boldsymbol{x}_{(2)}^{\mathsf{T}}\boldsymbol{x}_{(1)} \\ \vdots \\ \boldsymbol{x}_{(N-1)}^{\mathsf{T}}\boldsymbol{x}_{(1)} \end{bmatrix} + \rho \begin{bmatrix} \gamma_{(1)} \\ \gamma_{(2)} \\ \vdots \\ \gamma_{(N-1)} \end{bmatrix} - \begin{bmatrix} \boldsymbol{x}_{(1)}^{\mathsf{T}}\boldsymbol{x} \\ \boldsymbol{x}_{(2)}^{\mathsf{T}}\boldsymbol{x} \\ \vdots \\ \boldsymbol{x}_{(N-1)}^{\mathsf{T}}\boldsymbol{x} \end{bmatrix}.$$

The directional derivative of f at point $\boldsymbol{\beta}$ in the direction $\boldsymbol{e}_j$ is given by $\nabla f(\boldsymbol{\beta})^{\mathsf{T}}\boldsymbol{e}_j$ for $j \in \{1, 2, \ldots, (N-1)\}$. To ensure that the directional derivative of f at $\boldsymbol{e}_1$ towards any feasible direction (that is any direction that retains $\boldsymbol{\beta}$ within the unit simplex $\Delta^{(N-1)}$) is positive, it suffices to ensure that

$$\nabla f(\boldsymbol{e}_1)^{\mathsf{T}}(\boldsymbol{e}_j - \boldsymbol{e}_1) > 0, \quad \forall\, j \in \{2, \ldots, (N-1)\}. \tag{4.4.12}$$

The above condition holds if the following holds

$$\rho > \max_{j\in\{2,\ldots,(N-1)\}} \frac{(\boldsymbol{x}_{(1)} - \boldsymbol{x}_{(j)})^{\mathsf{T}}(\boldsymbol{x}_{(1)} - \boldsymbol{x}) + \varepsilon\gamma_{(1)}^2}{\gamma_{(j)} - \gamma_{(1)}}. \tag{4.4.13}$$

Eq. (4.4.11) is obtained by combining the above inequality with the requirement that $\rho \geq 0$. $\square$

4.4.4 Sufficient Condition for the Trivial Solution

Next, we show that Eq. (4.4.13) is a sufficient condition for the trivial solution $\boldsymbol{\beta}^\star = \boldsymbol{e}_1$.

Proposition 4.4.2. Assume that the nearest neighbour of $\boldsymbol{x}$ is unique, i.e. $\boldsymbol{x}_{(1)} \neq \boldsymbol{x}_{(j)}$

for $j \neq 1$. If the following holds

$$\rho > \max_{j \in \{2,\dots,(N-1)\}} \frac{(\boldsymbol{x}_{(1)} - \boldsymbol{x}_{(j)})^\top (\boldsymbol{x}_{(1)} - \boldsymbol{x}) + \varepsilon \gamma_{(1)}^2}{\gamma_{(j)} - \gamma_{(1)}}, \tag{4.4.14}$$

then the solution to Eq. (4.4.10) is given by $\boldsymbol{\beta}^\star = \boldsymbol{e}_1$. In addition, if for all $j \in \{2,\dots,(N-1)\}$ we have $\left(\boldsymbol{x}_{(j)} - \boldsymbol{x}_{(1)}\right)^\top \left(\boldsymbol{x} - \boldsymbol{x}_{(1)}\right) \leqslant 0$, then the solution to Eq. (4.4.10) is given by $\boldsymbol{\beta}^\star = \boldsymbol{e}_1$ for all $\rho > 0$.

Proof. For the first part of the proposition, if Eq. (4.4.14) holds, then for all j ($j \in \{2,\dots,(N-1)\}$) we have

$$\begin{aligned}
& \rho > \frac{(\boldsymbol{x}_{(1)} - \boldsymbol{x}_{(j)})^\top (\boldsymbol{x}_{(1)} - \boldsymbol{x}) + \varepsilon \gamma_{(1)}^2}{\gamma_{(j)} - \gamma_{(1)}} \\
\Leftrightarrow \quad & \boldsymbol{x}_{(j)}^\top \left(\boldsymbol{x}_{(1)} - \boldsymbol{x}\right) + \rho\gamma_{(j)} > \boldsymbol{x}_{(1)}^\top \left(\boldsymbol{x}_{(1)} - \boldsymbol{x}\right) + \varepsilon\gamma_{(1)}^2 + \rho\gamma_{(1)} \\
\Leftrightarrow \quad & \nabla f(\boldsymbol{e}_1)^\top \boldsymbol{e}_j > \nabla f(\boldsymbol{e}_1)^\top \boldsymbol{e}_1 \\
\Leftrightarrow \quad & \nabla f(\boldsymbol{e}_1)^\top (\boldsymbol{e}_j - \boldsymbol{e}_1) > 0.
\end{aligned}$$

The last line from above means that the directional derivative at $\boldsymbol{e}_1$ towards any other feasible direction within the unit simplex $\Delta^{(N-1)}$ is positive. Thus the solution to Eq. (4.4.10) is given by $\boldsymbol{\beta}^\star = \boldsymbol{e}_1$.

For the second part of the proposition, we first provide a geometric interpretation in Figure 4.4.1 for the meaning of the statement. In Figure 4.4.1, $\boldsymbol{x}$ is the point to be approximated and $\boldsymbol{x}_{(1)}$ is its nearest neighbour on the unit sphere. The bold black line is the perpendicular hyperplane of $\left(\boldsymbol{x} - \boldsymbol{x}_{(1)}\right)$, which is denoted by $\mathcal{H} = \left\{\boldsymbol{y} | \boldsymbol{y}^\top \left(\boldsymbol{x} - \boldsymbol{x}_{(1)}\right) = 0\right\}$.

Assume that all points apart from $\boldsymbol{x}$ and $\boldsymbol{x}_{(1)}$ lie on one side of the hyperplane $\mathcal{H}$, opposite the side which $\boldsymbol{x}$ resides in. That is, $\left(\boldsymbol{x}_{(j)} - \boldsymbol{x}_{(1)}\right)^\top \left(\boldsymbol{x} - \boldsymbol{x}_{(1)}\right) \leqslant 0$ for all $j \in \{2,\dots,(N-1)\}$. We can see that $\mathcal{H}$ is a supporting hyperplane for $\text{conv}(\mathcal{X} \backslash \{\boldsymbol{x}\})$. Consider for an arbitrary point $\boldsymbol{y} = Y\boldsymbol{\beta} \in \text{conv}(\mathcal{X} \backslash \boldsymbol{x})$, we have

$$\begin{aligned}
& \left(\boldsymbol{x} - \boldsymbol{x}_{(1)}\right)^\top \left(Y\boldsymbol{\beta} - \boldsymbol{x}_{(1)}\right) \\
&= \left(\boldsymbol{x} - \boldsymbol{x}_{(1)}\right)^\top \left[\sum_{j=1}^{N-1} \beta_j \left(\boldsymbol{x}_{(j)} - \boldsymbol{x}_{(1)}\right)\right]
\end{aligned}$$

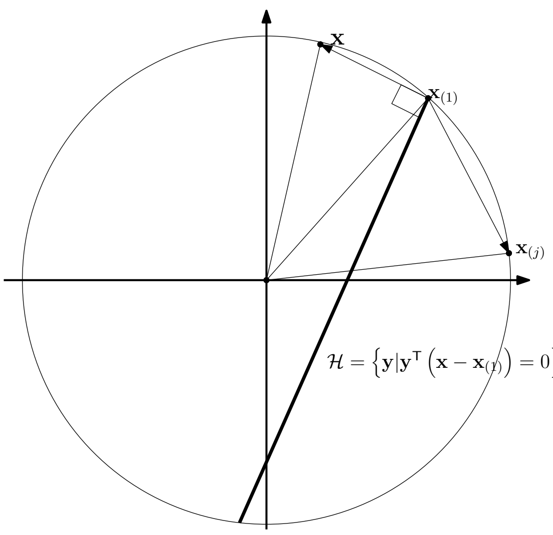

x
x(1)
x(j)
$\mathcal{H} = \left\{ \mathbf{y} | \mathbf{y}^{\mathsf{T}} \left(\mathbf{x} - \mathbf{x}_{(1)} \right) = 0 \right\}$

Figure 4.4.1: A geometric interpretation for when the trivial solution is obtained.

$$= \sum_{j=1}^{N-1} \beta_j \left(\boldsymbol{x} - \boldsymbol{x}_{(1)} \right)^{\mathsf{T}} \left(\boldsymbol{x}_{(j)} - \boldsymbol{x}_{(1)} \right)$$

$$\leqslant 0.$$

That is, all points apart from $\boldsymbol{x}$ and $\boldsymbol{x}_{(1)}$ lie on one side of the supporting hyperplane $\mathcal{H} = \left\{ \boldsymbol{y} | \boldsymbol{y}^{\mathsf{T}} \left(\boldsymbol{x} - \boldsymbol{x}_{(1)} \right) = 0 \right\}$. That is, $\left(\boldsymbol{x}_{(j)} - \boldsymbol{x}_{(1)} \right)^{\mathsf{T}} \left(\boldsymbol{x} - \boldsymbol{x}_{(1)} \right) \leqslant 0$. In this case, any linear combination of the column vectors in Y would be further away from $\boldsymbol{x}$ than using $\boldsymbol{x}_{(1)}$ itself as the approximation.

Therefore, the proposition says if $\left(\boldsymbol{x}_{(j)} - \boldsymbol{x}_{(1)} \right)^{\mathsf{T}} \left(\boldsymbol{x} - \boldsymbol{x}_{(1)} \right) \leqslant 0$ is satisfied for all $j \in \{2, \ldots, (N-1)\}$, then the trivial solution can be obtained for any $\rho > 0$. In general, for any data set $\mathcal{X}$, a trivial solution can be obtained for any ρ that satisfies Eq. (4.4.14). □

4.5 Constrained Clustering and Active Learning with WSSR (WSSR+)

In this section, we discuss how we can take full advantage of any available side information to improve the performance of WSSR. Depending on the type of side information, they can usually be divided into two categories: (a) the side information is given in the form of class labels; and (b) the side information is given in the form of pairwise 'must-link' and 'cannot-link' constraints. The former type can always be translated into the latter. For spectral-based subspace clustering methods in general, a good clustering result relies crucially on the quality of the affinity matrix. Given that it is natural to incorporate pairwise constraints into the affinity matrix, we translate all side information into pairwise 'must-link' and 'cannot-link' constraints.

We first discuss how to incorporate the constraint information into the WSSR problem formulation when the constraints are fixed and given, which corresponds to the setting of constrained clustering. Then we proceed to formulate an integrated constrained clustering and active learning framework when the labelling information of various points are queried sequentially over time.

4.5.1 Constrained Clustering

Our goal is to not only respect the constraints imposed by the side information, but also to use them as a guide to effectively uncover the correct cluster assignments of the unlabelled points. As is discussed in Section 4.2.2.5, Li et al. (2017) introduced a new subspace structured norm that encodes the constraint information into the optimisation objective. The constraint information is expressed in terms of the set of 'must-link' constraints $\mathcal{S}_M$, and the set of 'cannot-link' constraints $\mathcal{S}_C$. Inspired by their work, we modify the weight vector $\boldsymbol{\gamma}$ in Eq. (4.3.3) by incorporating the following information:

- Ψ: the N by N side information matrix where

$$\Psi_{ij} = \Psi_{ji} = \begin{cases} \exp(-1), & (i,j) \in \mathcal{S}_M, \\ \exp(0), & \text{no constraint information available}, \\ \exp(1), & (i,j) \in \mathcal{S}_C, \end{cases} \tag{4.5.1}$$

- Q: the $N \times K$ segmentation matrix $Q = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_N]^\mathsf{T}$, where Q_{ik} takes value one if $\boldsymbol{x}_i$ is assigned to cluster k, and zero otherwise. The matrix Q can be obtained from an initial data segmentation given by WSSR.

For each point $\boldsymbol{x}_i$, we obtain its updated weight vector $\boldsymbol{\gamma}^\star$ in Eq. (4.3.3) as follows

$$\gamma_j^\star = \gamma_j \Psi_{ij} + \frac{\alpha}{2} \left\| \boldsymbol{q}_i - \boldsymbol{q}_j \right\|_2^2 . \tag{4.5.2}$$

The first term in Eq. (4.5.2) says that if the pairwise relationship between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is known, then their pairwise dissimilarity gets reduced or increased depending on whether they have a 'must-link' or 'cannot-link' relationship. The second term in Eq. (4.5.2) is a measure of the agreement between the cluster labels of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, in which α is a penalty parameter between zero and one that reflects the level of faith we have in the segmentation matrix Q. A simple heuristic to determine α is to set it to be the percentage of points that have been queried so far. When the pairwise relationship between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is 'must-link', the second term in Eq. (4.5.2) vanishes. When the pairwise relationship between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is 'cannot-link', the second term becomes α.

By solving the WSSR problem with this updated weight vector $\boldsymbol{\gamma}^\star$, we can obtain an updated data affinity matrix. However, there is no guarantee that all constraints would be respected by solely applying spectral clustering to the updated affinity matrix. We ensure the satisfaction of all constraints by additionally applying the K-Subspace Clustering with Constraints (KSCC) algorithm that is introduced in Peng and Pavlidis (2019). We call this constrained version of WSSR as *Weighted Sparse Subspace Representation plus (WSSR+)*. The procedural form of WSSR+ can be found in Algorithm 9.

4.5.2 Active Learning

So far, we have discussed how to incorporate the constraint information when they are already given. However, there is no guarantee that the given set of constraints are the most informative as compared to a different set of constraints. We consider the scenario where we are able to sequentially query the labels for certain informative points, then use the queried information to assist clustering. This iterative process of querying external information and improving the cluster performance is called *active learning* (Settles, 2009).

Algorithm 9: WSSR with Constraints (WSSR+)

```
Input  : Initial cluster labels obtained from WSSR
         The set of ‘must-link’ and ‘cannot-link’ constraints: $\mathcal{S}_M, \mathcal{S}_C$
         Penalty parameter: $\alpha$
         Number of nearest neighbours: $k$
For $\boldsymbol{x} \in \mathcal{X}$:
1. Compute the updated weight vector $\boldsymbol{\gamma}^\star$ according to Eq. (4.5.2)
2. Normalise and stretch each column vector in $Y$
3. Solve the WSSR problem in Eq. (4.3.3) to obtain the coefficient vector $\boldsymbol{\beta}$
End
- Combine all $\boldsymbol{\beta}$s to obtain the coefficient matrix $B \in \mathbb{R}^{N \times N}$ according to
  Eq. (4.2.2)
- Apply normalised cut spectral clustering (Ng et al., 2002) to the data affinity
  matrix $A = \frac{1}{2}\left(|B| + |B|^\mathsf{T}\right)$ to obtain $K$ clusters
- Enforce the constraint information using KSCC and obtain the updated cluster
  labels
```

We further propose a unified framework that is composed of two iterative stages. In the first stage, the class labels of a few informative points are queried according to the active strategy proposed in Peng and Pavlidis (2019). Then we translate the class labels into sets of pairwise ‘must-link’ constraints $\mathcal{S}_M$ and ‘cannot-link’ constraints $\mathcal{S}_C$. In the second stage, the constraints are satisfied and the cluster labels are updated using WSSR+. We provide the procedural form for this unified framework in Algorithm 10.

Algorithm 10: WSSR+ with Active Learning (WSSR++)

```
Input  : Initial cluster labels
         Penalty parameter: $\alpha$
repeat
    1. Query informative points using the proposed active query strategy in Peng
       and Pavlidis (2019)
    2. Transform the queried class labels into pairwise constraints $\mathcal{S}_M$ and $\mathcal{S}_C$
    3. Solve the constrained clustering problem using WSSR+ (as detailed in
       Algorithm 9)
until *A query budget is reached*
```

4.6 Experiments on Synthetic Data

Previously, we have shown that the WSSR sub-problem without the non-negativity constraint (WSR) can be solved analytically by solving a system of linear equations. In this section, we conduct experiments on synthetic data to evaluate the performance of both WSR and WSSR under various subspace settings. We compare to the following state-of-the-art spectral-based subspace clustering methods: SSC (Elhamifar and Vidal, 2013), S3C (Li and Vidal, 2015), ASSC (Li et al., 2018a), SSC-OMP (You et al., 2016), LSR (Lu et al., 2012), and SMR (Hu et al., 2014).

The data matrix can be expressed as $Y = X + E$, where X is the noise-free data and E the noise component. In order to generate the noise-free data matrix $X \in \mathbb{R}^{N \times P}$, we need to generate each sub-matrix $X_k \in \mathbb{R}^{n_k \times P}$ ($k \in \{1, \ldots, K\}$) for each subspace individually and concatenate them to form X. To generate the basis vectors for each subspace $\mathcal{S}_k$, we first generate a $P \times q$ matrix $B_k^\star$ whose entries come from the standard Normal distribution with $\mathcal{N}(0, 1)$. Then we orthogonalise the columns of $B_k^\star$ to obtain the matrix B_k whose columns correspond to the basis vectors for the subspace. The noise-free sub-matrix X_k can thus be obtained as

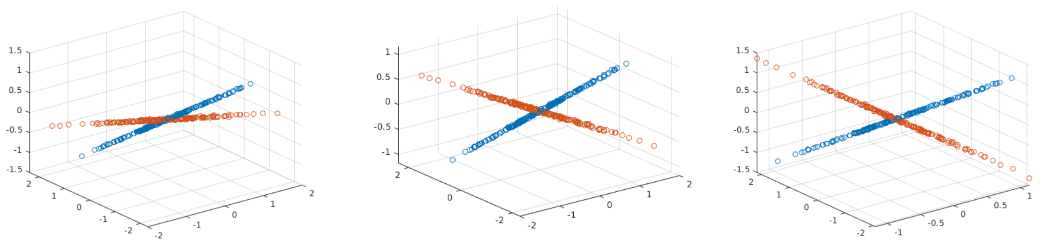
$$X_k = (B_k C_k)^\mathsf{T}, \tag{4.6.1}$$

where $C_k \in \mathbb{R}^{q \times n_k}$ is the coefficient matrix, whose entries are also sampled from the standard Normal distribution. Each column in C_k corresponds to the coefficient vector of a point along the q subspace dimensions. Each entry in the noise data matrix E is generated from standard Normal distribution $\mathcal{N}(0, \sigma^2)$, with zero mean and variance σ^2.

Performance results of various methods are compared in terms of the following three aspects: angles between subspaces, noise level, and dimension of the subspaces. We have the following hypotheses regarding these three aspects. First, cluster performance from different algorithms should improve with the increase in the angles between subspaces. Similarly, performance usually suffer with the increase of noise levels whilst everything else remains the same. Given a fixed full data dimension, the clusters are more likely to overlap with each other with the increase of subspace dimensions. Thus performance are likely to decrease as a result. Finally, we test how sensitive the performance of WSSR is to the key parameter – neighbourhood size k.

4.6.1 Varying Angles between Subspaces

In this set of experiments, we generate data from two one-dimensional subspaces embedded in a three-dimensional space. Each cluster contains 200 data points drawn from one of the subspaces. In addition, additive Gaussian noise with standard deviation $\sigma = 0.01$ is added to the data uniformly. We vary the angles between the two subspaces θ to be between 10 and 60 degrees, and evaluate the performance of various algorithms under each setting. Three illustrations of the data with varying angles between subspaces are shown in Figure 4.6.1.



(a) 20 degrees. (b) 40 degrees. (c) 60 degrees.

Figure 4.6.1: Three illustrations of data from two clusters under varying angles between the two.

The default settings are adopted for all subspace clustering algorithms that we provide comparisons to. For WSSR and WSR, we use $k = 10$ and $\rho = 0.01$, which is the same as SSC-OMP and SMR. Both of these algorithms have a parameter that controls the maximum number of points to consider in the sparse representation. Performance results as evaluated by clustering accuracy are reported in Table 4.1. We highlight the best results in bold, and underline the second best results.

It can be seen that WSSR achieves the best performance across all settings, whilst increasing its performance steadily with the increase of the angles between subspaces.

Despite the lack of the non-negativity constraint, WSR also has a relatively close performance to that of WSSR. SSC and S3C achieve strong performance across all scenarios as well. It is worth noting that the performance of ASSC is noticeably worse than other algorithms, which could be explained by the fact that all clusters come from linear subspaces. The performance of SMR is a close second to WSSR in five out of six scenarios. This could be attributed to its affinity to WSSR, as SMR applies the Frobenius norm on the error matrix and it also makes use of k nearest neighbours.

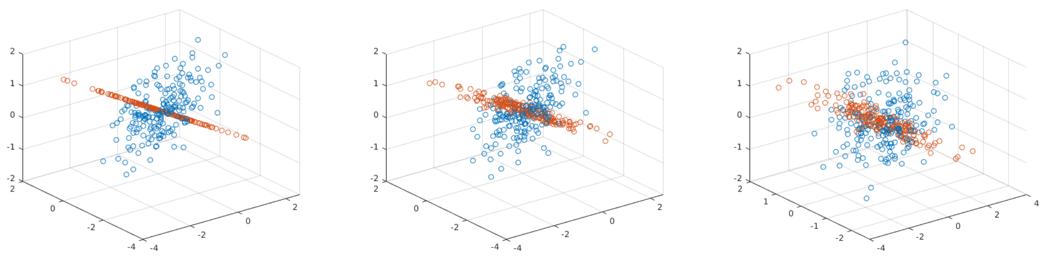
	$\theta = 10$	$\theta = 20$	$\theta = 30$	$\theta = 40$	$\theta = 50$	$\theta = 60$
WSSR	**0.978**	**0.973**	**0.993**	**0.993**	**0.990**	**0.993**
WSR	0.965	0.918	0.990	0.973	0.988	0.978
SSC	0.943	0.815	0.990	0.950	0.985	**0.993**
S3C	0.963	**0.973**	0.990	0.970	0.983	**0.993**
ASSC	0.520	0.570	0.570	0.568	0.510	0.555
SSC-OMP	0.863	0.893	0.848	0.823	0.528	0.813
LSR	0.898	0.878	0.900	0.873	0.940	0.930
SMR	0.968	0.960	0.990	0.975	0.978	0.988

Table 4.1: Accuracy of various subspace clustering algorithms on synthetic data with varying angles between subspaces.

4.6.2 Varying Noise Levels

Next, we explore the effect of various noise levels on cluster performance. Again we generate data from two subspaces, each containing 200 data points. The angle between two subspaces is set to be 60 degrees, so that the angle between subspaces does not play a big role in determining the cluster performance. One of the subspaces is one-dimensional, and the other is two-dimensional. This difference to the previous set of experiments is to increase the intersection between the two subspaces as the noise level increases. Additive Gaussian noise with zero mean and standard deviation σ is added to the data uniformly, in which σ ranges from 0.0 to 0.5. A visualisation of the data under three noise levels can be seen in Figure 4.6.2.

The parameter settings for all algorithms remain the same as before, and the performance results are reported in Table 4.2. Firstly, clustering accuracy of almost all



(a) Noise-free. (b) Noise level $\sigma = 0.2$. (c) Noise level $\sigma = 0.5$.

Figure 4.6.2: Three illustrations of data from two clusters with varying noise levels.

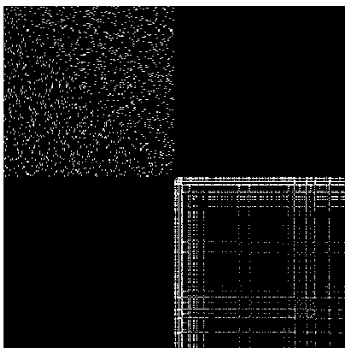
algorithms decreases with the increase of noise levels. Secondly, most algorithms have close to or exactly perfect clustering accuracy in the noise-free scenario. Very poor performance from ASSC can be observed across all noise levels, for reasons explained in the previous set of experiments.

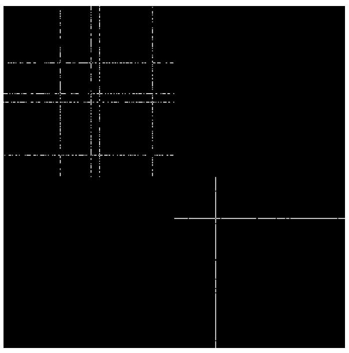
	$\sigma = 0.0$	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.3$	$\sigma = 0.4$	$\sigma = 0.5$
WSSR	**1.000**	**0.970**	**0.945**	**0.883**	**0.815**	**0.745**
WSR	**1.000**	0.845	0.690	0.558	0.633	0.505
SSC	**1.000**	0.633	0.503	0.513	0.508	0.555
S3C	0.980	0.685	0.575	0.608	0.523	0.593
ASSC	0.605	0.530	0.553	0.558	0.503	0.543
SSC-OMP	**1.000**	0.730	0.575	0.510	0.518	0.530
LSR	**1.000**	0.943	0.900	0.838	0.788	0.735
SMR	0.980	0.935	0.868	0.810	0.775	0.705

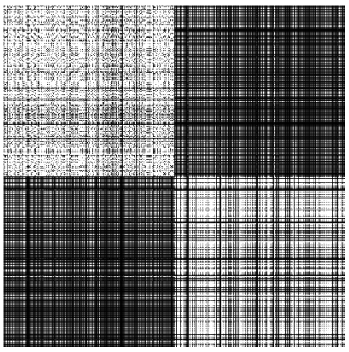
Table 4.2: Accuracy of various subspace clustering algorithms on synthetic data with varying noise levels.

When we compare the performance between WSSR and WSR, we see that the gaps here are much larger that what have been observed in the previous set of experiments. Indeed, the gap gets wider as the noise level increases. This demonstrates the advantage of WSSR over WSR in the presence of noise. It is also worth noting that all SSC-based methods (SSC, S3C, ASSC, SSC-OMP) yield poor performance in the presence of varying levels of noise. In comparison, the two methods (LSR and SMR) that both use the Frobenius norm on the error matrix have favourable performance. However, neither of them have a sparsity inducing term in the objective function. That is, the same reason for why the performance of WSR is inferior to WSSR can also be applied to these two methods.

Next, we further investigate the reason behind the slightly less than perfect performance of S3C and SMR, and the poor performance of ASSC in the noise-free scenario. Shown in Figure 4.6.3 are the affinity matrices for WSSR, S3C, SMR, and ASSC. It is clear to see that the entries in the affinity matrix of WSSR are sparse yet the block-diagonal structure is very clear. The affinity matrix of S3C also exhibits a block-diagonal structure and is very sparse. However, the majority of the non-zero entries are concentrated on a few key data points. Such an unbalanced affinity matrix can lead to undesirable spectral clustering performance. The affinity matrix of both SMR and ASSC are very dense. However the block diagonal structure can still be easily detected in the affinity matrix of SMR, whereas the entries in ASSC are more evenly spread out. Therefore, it is not surprising that the performance of ASSC is much worse than that of the other methods.


(a) WSSR.


(b) S3C.


(c) SMR.

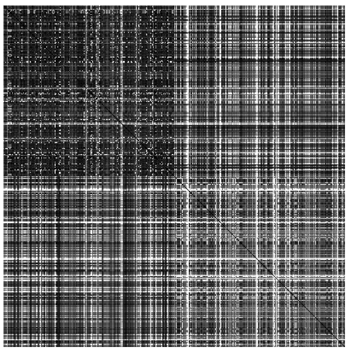

(d) ASSC.

Figure 4.6.3: Visualisation of the affinity matrix in the noise-free scenario.

4.6.3 Varying Subspace Dimensions

Another aspect we would like to investigate is how clustering performance changes with the increase of subspace dimensions q under a fixed ambient space dimension P. Intuitively, one would expect the performance to get worse as q increases. This is because the intersection among clusters is likely to increase as the subspace dimension grows. In this set of experiments, we fix the ambient space dimension to be 20 and allow the subspace dimension to vary between 2 and 16. Data are generated from 4 subspaces, and all subspaces have equal subspace dimension. Additive noise with $\sigma = 0.01$ is added to the data uniformly. We set the neighbourhood size k in WSSR and WSR to be 50, and keep $\rho = 0.01$. The default parameter settings are adopted for all other methods. The performance results of various methods are reported in Table 4.3.

	$q=2$	$q=4$	$q=6$	$q=8$	$q=10$	$q=12$	$q=14$	$q=16$
WSSR	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.991**	**0.874**
WSR	**1.000**	**1.000**	**1.000**	<u>0.998</u>	<u>0.995</u>	0.953	0.688	0.363
SSC	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	<u>0.964</u>	0.728
S3C	<u>0.999</u>	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.991**	<u>0.809</u>
ASSC	0.845	0.998	**1.000**	**1.000**	**1.000**	<u>0.994</u>	0.954	0.599
SSC-OMP	0.414	**1.000**	<u>0.999</u>	0.993	0.966	0.913	0.475	0.321
LSR	0.861	<u>0.999</u>	**1.000**	**1.000**	<u>0.995</u>	0.986	0.936	0.518
SMR	<u>0.999</u>	0.998	0.968	0.981	0.823	0.469	0.335	0.323

Table 4.3: Accuracy of various subspace clustering algorithms on synthetic data with varying subspace dimensions.

We observe that WSSR achieves the best performance across all settings, though the accuracy worsens when q is greater than 12. As for WSR, we observe a more obvious downward trend in the clustering accuracy as q increases. The gap in the performance between WSSR and WSR also gets larger with the increase of q. This agrees with our previous interpretation that WSR is not as resilient as WSSR in the presence of noise. As previously mentioned, the increase of subspace dimensions likely increases the intersection between subspaces hence induces more noise.

When we move our focus to the performance of SSC-based methods, we see that both SSC and S3C also have near perfect clustering accuracy for q up to 12. Then the results get slightly worse for higher values of q. For q less than 16, the results of ASSC are much better than in the previous set of varying noise experiments. This shows that ASSC has the subspace recovery ability when the noise level is small. This ability does not seem to be as good as that of WSR, which shares a similar formulation with ASSC. SSC-OMP has worse accuracy than its base method SSC, which could be attributed to its default neighbourhood size. LSR and SMR maintain fairly good performance throughout, with the same decreasing trend of clustering accuracy.

To make a fair comparison, we further inspect the sensitivity of WSSR, SSC-OMP, and SMR to the choice of neighbourhood size k, a key parameter that all methods rely on. For each value of q from 2 to 20, we conduct 20 random trials for each of the three methods. Within each random trial, the neighbourhood size k is randomly chosen from the range $[2, 100]$. Figure 4.6.4 visualises the median clustering accuracy for each method in solid lines. In addition, the range between minimum and maximum accuracy are coloured in shaded areas.

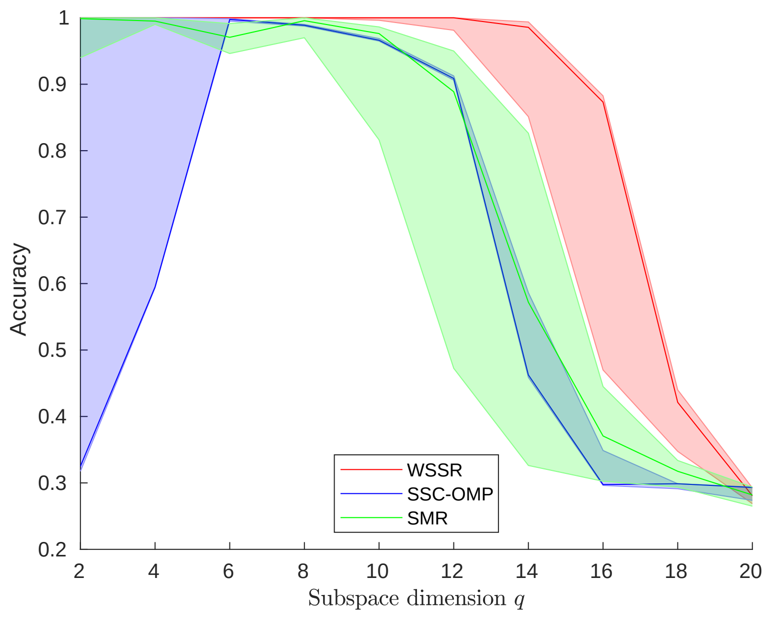


Figure 4.6.4: Performance results of three methods across varying subspace dimensions. For each subspace dimension q, 20 trials are conducted with randomly chosen neighbourhood size k.

From the performance results of WSSR, we can see that the accuracy stays strong and stable for q up to 10. That is, there is barely any variability in the clustering accuracy irrespective of the choice of k. However, an increase in the performance variability can be seen with the increase of q. With that said, it is worth noting that the line for median clustering accuracy stays very much towards the upper end within the range of accuracy values. Whereas the opposite phenomenon can be observed for SSC-OMP, in which the median performance always stays at the lower end within the range of accuracy values. Compared to WSSR, a smaller range of variability can be seen for SSC-OMP when q is large. However, the performance variability is extremely large when q is smaller than 6. Finally when we move our focus to the performance of SMR, we observe the widest range of variability across all three methods for large values of q. Though when q is small, the performance is fairly stable under varying choices of the neighbourhood size.

4.7 Experiments on Real Data

In this section we discuss the experimental performance of WSSR with and without side information. We will evaluate the following aspects of our proposed constrained subspace clustering framework:

- In the absence of any side information, how does WSSR compare with other state-of-the-art subspace clustering methods?
- When there is a fixed amount of side information, how does the performance of WSSR+ compare with other constrained clustering methods?
- When the side information can be actively queried sequentially over time, do we see any performance improvement as compared to the previous scenario?

4.7.1 WSSR Experiments on MNIST Data

In this section, we conduct experiments comparing WSSR with other state-of-the-art subspace clustering algorithms on the MNIST handwritten digits data (LeCun et al., 1998). This database contains greyscale images of handwritten digits numbered from 0 to 9. It has been previously used in You et al. (2016) to demonstrate the effectiveness of the proposed SSC-OMP subspace clustering algorithm. It is one of the best performing algorithms in subspace clustering. As such, SSC-OMP is one of the algorithms we provide comparisons to in our experiments. In addition, we also compare to other state-of-the-art subspace clustering algorithms including SSC (Elhamifar and Vidal, 2013), ASSC (Li et al., 2018a), S3C (Li and Vidal, 2015), LSR (Lu et al., 2012), SMR (Hu et al., 2014), and FGNSC (Yang et al., 2019a).

The original MNIST data set contains 60,000 points from 10 clusters (digits 0 to 9), with the full data dimension $P = 3472$. We conduct two sets of experiments on the data. One set of experiments investigates the effects of number of clusters K on the cluster performance of various algorithms. We randomly select $K \in \{2, 3, 5, 8, 10\}$ clusters out of the 10 digits. Each cluster contains 100 randomly sampled points, and the full data are projected to dimension 200 using PCA. Another set of experiments follows the same experimental design in You et al. (2016), and explores the effects of the total number of data points N on the cluster performance. In this set of experiments, the data are processed in the same way as in You et al. (2016). We randomly select $N_k \in \{50, 100, 200, 400, 600\}$ points from each cluster out of all 10 clusters. The full data are then projected onto dimension 500 using PCA.

We follow the default parameter settings for SSC ASSC, and SSC-OMP. There are two versions of LSR: LSR-1 is the standard version of the LSR problem with the constraint requiring the diagonal entries of the coefficient matrix have to be zero; and LSR-2 is the unconstrained version of LSR. We use LSR-1 in our experiments, as LSR-2 does not include the constraint that the diagonal entries of the representation matrix have to be zeroes. The default parameter setting for SMR uses $k = 4$ for the k-nearest neighbour graph. We adopt the default setting of FGNSC, which uses SMR as the base algorithm to obtain the initial affinity matrix. In WSSR, we set $k = 10$ and $\rho = 0.01$. That is we consider a 10-nearest neighbourhood for each data point, which is the same as the default setting in SSC-OMP.

We conduct 20 trials for each method that we experiment with, and the performance

	$K = 2$		$K = 3$		$K = 5$		$K = 8$		$K = 10$	
	Med	Std	Med	Std	Med	Std	Med	Std	Med	Std
WSSR	**1.00**	0.01	**1.00**	0.01	**0.99**	0.01	**0.98**	0.01	**0.98**	0.02
WSR	**1.00**	0.05	0.98	0.02	0.81	0.08	0.79	0.05	0.81	0.02
SSC	<u>0.99</u>	0.03	0.91	0.05	0.80	0.08	0.78	0.03	0.81	0.02
S3C	<u>0.99</u>	0.02	0.97	0.08	0.79	0.08	0.82	0.04	0.81	0.03
ASSC	<u>0.99</u>	0.01	0.97	0.02	<u>0.95</u>	0.05	0.87	0.06	0.84	0.04
SSC-OMP	0.98	0.01	0.97	0.02	<u>0.95</u>	0.04	<u>0.90</u>	0.04	<u>0.86</u>	0.03
LSR	0.98	0.11	0.68	0.11	0.86	0.07	0.82	0.04	0.78	0.02
SMR	<u>0.99</u>	0.04	<u>0.98</u>	0.02	<u>0.95</u>	0.06	0.86	0.05	0.83	0.03
FGNSC	<u>0.99</u>	0.06	<u>0.98</u>	0.03	0.83	0.09	0.84	0.04	0.85	0.04

Table 4.4: Median clustering accuracy along with the standard deviations on the MNIST handwritten digits data across 20 trials with $P = 200$.

results as measured by both the median clustering accuracy and the standard deviation are reported in Table 4.4. It can be seen that WSSR achieves the best performance across all settings. In particular, it achieves perfect clustering accuracy for both $K = 2$ and 3. It is also worth noting that the performance variability is relatively small as compared to other methods. As is expected, WSR has similar performance to WSSR when K is small. However the gap between these two enlarges with the increase of K.

For the competing algorithms, we see that a few algorithms have excellent performance when K is small. However the performance degrades in general with the increase of K. Both SSC and S3C have very similar performance to each other, with S3C having slightly higher accuracy scores on two scenarios. The performance of ASSC is higher than both of the previous two, and has similar performance to that of SSC-OMP. However, SSC-OMP has an obvious edge over the other SSC-based methods when K is large. This gives favourable evidence to the benefit of focusing on a nearest neighbourhood.

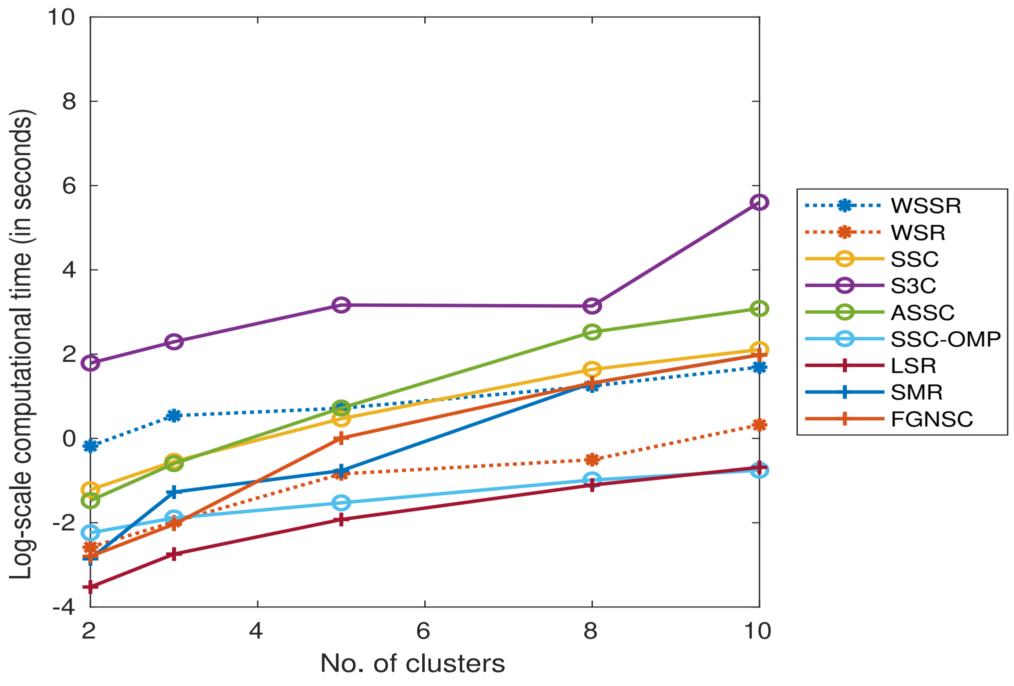
The median performance of LSR seems to be the worst out of all methods, and is the most variable especially when K is small. SMR and FGNSC have the same level of performance when $K = 2$ and 3, which is not surprising given that the affinity matrix of FGNSC is adapted from that of SMR. However the performance of FGNSC is less stable for larger values of K. It has been observed in our experiments that the performance of FGNSC is highly sensitive to the parameter values.

	$N = 500$		$N = 1000$		$N = 2000$		$N = 4000$		$N = 6000$	
	Med	Std	Med	Std	Med	Std	Med	Std	Med	Std
WSSR	**0.96**	0.03	**0.98**	0.00	**0.98**	0.00	**0.99**	0.00	**0.99**	0.00
WSR	0.74	0.03	0.79	0.04	0.83	0.02	0.85	0.02	0.86	0.01
SSC	0.78	0.04	0.81	0.03	0.82	0.02	0.83	0.01	0.84	0.01
S3C	0.78	0.04	0.82	0.04	0.82	0.03	0.83	0.02	0.83	0.02
ASSC	0.82	0.04	0.85	0.04	0.83	0.02	0.82	0.01	0.83	0.01
SSC-OMP	$\underline{0.83}$	0.04	$\underline{0.88}$	0.03	$\underline{0.91}$	0.02	$\underline{0.92}$	0.03	0.92	0.04
LSR	0.66	0.04	0.74	0.03	0.78	0.02	0.79	0.01	0.80	0.01
SMR	0.76	0.03	0.82	0.05	0.88	0.04	0.86	0.04	0.92	0.04
FGNSC	0.77	0.03	0.85	0.03	0.87	0.04	0.87	0.04	$\underline{0.97}$	0.02

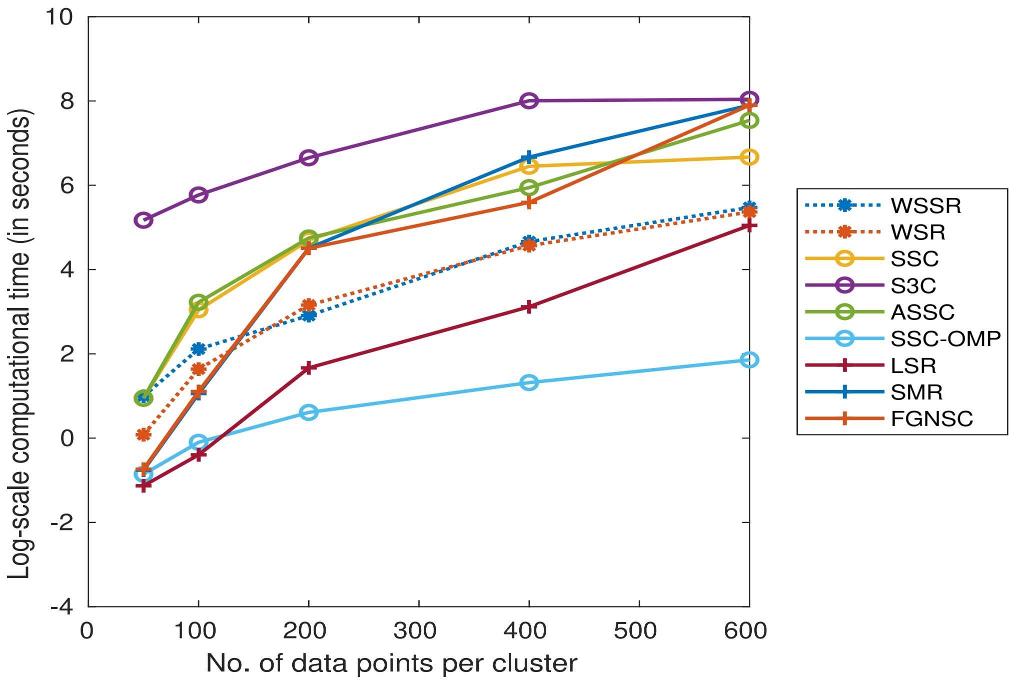
Table 4.5: Median clustering accuracy along with the standard deviations on the MNIST handwritten digits data across 20 trials with $P = 500$.

The performance results for the second set of experiments with varying N are reported in Table 4.5. WSSR is again the best performing algorithm across all settings, and indeed the most stable of all as well. It can be seen that SSC-OMP is the algorithm with the second best performance in most settings. The performance of SSC-OMP reported here is consistent with what is reported in (You et al., 2016). In general, the cluster performance improves with increasing number of points. At the same time the performance for SSC, ASSC, and LSR also become more stable as N increases, which is not the case for SMR. With that said, FGNSC provides performance improvement in addition to the performance of SMR in most scenarios.

All of the above experiments are run on a cloud computing machine with 5 CPU cores and 15GB of RAM. For each of the two sets of experiments, we present a comparison of the median computational times in log-scale for different algorithms in Figure 4.7.1. When comparing the computational times between Figure 4.7.1 (a) and 4.7.1 (b), it can be seen that the times in (a) when $K = 10$ is at about the same level as those in (b) when $N_k = 100$. This is to be expected because both scenarios involve the use of 1000 points. When comparing the computational times within each figure, it is clear that S3C is the most computational intensive out of all methods. SSC and ASSC have similar computational times because they are based on the same optimisation framework. SSC-OMP and LSR are the most efficient out of all methods. SSC-OMP is suitable for large-scale problems, and LSR has a closed-form solution. With that said, the computational times of both WSSR and WSR are favourable as compared to other competing methods.



(a) Varying K.



(b) Varying N.

Figure 4.7.1: Median running times (in log-scale of seconds) of different algorithms on the MNIST handwritten digits data.

4.7.2 WSSR Experiments on USPS Data

In this experiment, we evaluate the cluster performance of difference subspace clustering methods on the USPS digits data (Hull, 1994). It is another popular benchmark data set that has been used to demonstrate the effectiveness of several subspace clustering methods (Hu et al., 2014; Yang et al., 2019a). There are 9298 images of handwritten digits that range from 0 to 9, and each image contains 16×16 pixels. We follow the exact same experimental settings as in Hu et al. (2014), which uses the first 100 images from each digit.

	$K = 2$		$K = 3$		$K = 5$		$K = 8$		$K = 10$	
	Med	Std	Med	Std	Med	Std	Med	Std	Med	Std
WSSR	**1.00**	0.01	**0.99**	0.01	**0.98**	0.01	**0.97**	0.00	**0.97**	0.00
WSR	0.98	0.02	0.97	0.08	0.78	0.07	0.81	0.05	0.83	0.02
SSC	0.96	0.03	0.85	0.12	0.63	0.14	0.60	0.05	0.51	0.00
S3C	0.97	0.02	0.83	0.13	0.69	0.09	0.62	0.05	0.51	0.01
ASSC	0.97	0.02	0.83	0.12	0.61	0.14	0.62	0.06	0.58	0.00
SSC-OMP	0.95	0.04	0.74	0.11	0.42	0.09	0.41	0.09	0.35	0.01
LSR	0.71	0.14	0.63	0.13	0.65	0.05	0.55	0.04	0.53	0.01
SMR	$\underline{0.99}$	0.04	$\underline{0.98}$	0.02	$\underline{0.95}$	0.06	$\underline{0.86}$	0.05	0.83	0.03
FGNSC	$\underline{0.99}$	0.02	$\underline{0.98}$	0.05	$\underline{0.95}$	0.06	0.83	0.05	$\underline{0.85}$	0.02

Table 4.6: Median clustering accuracy along with the standard deviations on the USPS data across 20 trials.

We investigate the performance of various algorithms under varying number of clusters K. All experiments are conducted for 20 trials, and we report both the median and standard deviation of the clustering accuracy. For K from 2 to 8, we randomly sample data from K digits out of 10. Therefore the variability in the cluster performance comes from both the variability in the subset of the data, and the variability of the corresponding algorithm. For $K = 10$, we use the same data set with 1000 images across all trials. In this case, the standard deviation reflects only the variability of the algorithms.

The performance behaviours of WSSR and WSR on the USPS data are similar to their performance on the MNIST data. It can be seen that the performance of WSR is much more variable than that of WSSR for $K = 3$, 5, and 8. The performance of SSC , AS3C, and ASSC are fairly similar to each other. Similar to WSR, these three methods also have very variable performance results in most cases. It is worth noting that the performance results of SSC-OMP are much worse than its three SSC-based counterparts in most scenarios. This could be due to its sensitivity to the neighbourhood size parameter k, as we have demonstrated before in Section 4.6.3. The accuracy scores of LSR are mediocre across all settings, and it also seems to have a much higher performance variability when K is small. Both SMR and FGNSC have excellent performance when K is small, however the supposedly additional benefit of FGNSC is not exhibited here.

4.7.3 WSSR+ Experiments

In this section, we evaluate the additional performance improvement that our proposed WSSR+ constrained clustering and active learning framework could bring. Constrained clustering and active learning experiments are conducted on both the MNIST data and the USPS data. For each data set with varying number of clusters K, we first compare the performance of various constrained clustering methods under varying proportions of available class information. Each experiment is repeated across 20 trials where a certain proportion $p\%$ of class information is sampled at random.

Alongside this, we also report the performance under the same proportion $p\%$ where the class information is actively queried over time using the strategy as proposed in Peng and Pavlidis (2019). This active strategy queries the most informative point according to the underlying subspace structure. An informative point is not only potentially misclassified, but can also lead to the correct labelling assignment of other points upon being queried. The active learning framework is composed of three stages. In the first stage, we utilise the active strategy to query labelling information sequentially. In the second stage, we apply WSSR+ to the data with the incorporation of side information to obtain the cluster assignments. Finally, we use the K-subspace clustering with constraints (KSCC) algorithm as proposed in Peng and Pavlidis (2019) to enforce the constraint satisfaction. We compare WSSR+ with two other state-of-the-art constrained clustering methods: Constrained Spectral Partitioning (CSP) (Wang et al., 2014) and Partition Level Constrained Clustering (PLCC) (Liu et al., 2018).

Experimental settings. The initial clustering performance without any side information is produced by WSSR. For fair comparison, we use the affinity matrix A produced by WSSR as the initial input for all competing constrained methods. There is one tuning parameter λ in PLCC which controls how much weight should be put on the side information. Although it is recommended in Liu et al. (2018) to set λ to be above 10,000 for stable performance, we have found in our experiments that this has a detrimental influence on the resulting clustering performance. As such, we conduct 20 random searches for λ in the range (0,1) and choose the value that gives the best clustering accuracy for PLCC. No extra parameter needs to be set for CSP. The constrained clustering and active learning results on both the MNIST data and the USPS data are reported in Table 4.7 and Table 4.8 respectively.

Since we provide comparison across all three methods under varying number of clusters, under varying percentage of side information, and under both active learning and constrained clustering, it is difficult to highlight the best performing result for each scenario. Instead, we provide sufficient discussion of the results in the remainder of the section.

K	Pct.	WSSR	WSSR+			PLCC			CSP		
			AL	Med	Std	AL	Med	Std	AL	Med	Std
2	10%	1.00	1.00	1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00
	20%		1.00	1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00
	30%		1.00	1.00	0.00	0.94	1.00	0.00	1.00	1.00	0.00
3	10%	1.00	1.00	1.00	0.00	1.00	1.00	0.00	0.68	0.99	0.00
	20%		1.00	1.00	0.00	1.00	1.00	0.00	0.68	1.00	0.00
	30%		1.00	1.00	0.00	1.00	0.52	0.01	0.68	1.00	0.00
5	10%	1.00	1.00	1.00	0.00	1.00	0.99	0.00	0.79	0.44	0.17
	20%		1.00	1.00	0.00	1.00	0.99	0.00	0.80	0.64	0.12
	30%		1.00	1.00	0.00	0.98	0.45	0.01	0.80	0.99	0.07
8	10%	0.98	0.99	0.98	0.00	0.86	0.62	0.06	0.97	0.44	0.17
	20%		0.99	0.98	0.00	0.86	0.55	0.05	0.99	0.65	0.11
	30%		0.99	0.99	0.00	0.85	0.80	0.05	0.99	0.89	0.09
10	10%	0.98	0.98	0.98	0.00	0.79	0.78	0.05	0.97	0.38	0.13
	20%		0.99	0.99	0.00	0.77	0.82	0.06	0.98	0.52	0.11
	30%		0.99	0.99	0.00	0.80	0.46	0.02	0.88	0.78	0.10

Table 4.7: Clustering accuracy of various constrained clustering methods on the MNIST data. The initial affinity matrix for all methods is produced by WSSR.

K	Pct.	WSSR	WSSR+			PLCC			CSP		
			AL	Med	Std	AL	Med	Std	AL	Med	Std
2	10%	1.00	1.00	1.00	0.00	1.00	1.00	0.00	1.00	0.99	0.01
	20%		1.00	1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00
	30%		1.00	1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00
3	10%	0.99	0.99	0.99	0.00	1.00	0.99	0.00	0.99	0.98	0.13
	20%		0.99	0.99	0.00	1.00	0.99	0.00	1.00	0.99	0.00
	30%		1.00	0.99	0.00	1.00	0.52	0.02	1.00	0.99	0.00
5	10%	0.97	0.98	0.97	0.00	0.97	0.97	0.00	0.98	0.35	0.11
	20%		0.98	0.97	0.00	0.98	0.98	0.01	0.70	0.52	0.10
	30%		0.98	0.98	0.00	0.89	0.45	0.01	0.98	0.80	0.13
8	10%	0.97	0.97	0.97	0.00	0.82	0.54	0.07	0.80	0.31	0.12
	20%		0.98	0.97	0.00	0.81	0.50	0.04	0.96	0.47	0.11
	30%		0.98	0.98	0.00	0.79	0.75	0.03	0.97	0.73	0.08
10	10%	0.97	0.97	0.97	0.00	0.65	0.85	0.05	0.97	0.35	0.13
	20%		0.97	0.97	0.00	0.73	0.85	0.06	0.96	0.49	0.11
	30%		0.98	0.98	0.00	0.73	0.48	0.01	0.88	0.74	0.10

Table 4.8: Clustering accuracy of various constrained clustering methods on the USPS data. The initial affinity matrix for all methods is produced by WSSR.

In Table 4.7, we can see that the initial cluster accuracy for K from 2 to 5 without any available class information is already 1. In this case, we would like to inspect whether various constrained clustering algorithms would retain the perfect clustering given additional class information. As it can be seen, WSSR+ is able to retain the perfect clustering accuracy irrespective of the level of side information available. Whereas we see a performance drop in some cases for PLCC and CSP. In general, we can see the benefit of active learning as opposed to random sampling when acquiring the additional class information. Most methods exhibits higher clustering accuracy under the active learning scheme.

Similar behaviours can be observed in Table 4.8 where various methods are applied on the USPS data. Perfect clustering accuracy is achieved when $K = 2$, we further test the stability of various methods with additional class information. It can be seen that all methods maintain perfect cluster performance when active learning is used. However, the accuracy of CSP with 10% randomly sampled labelling information is slightly less than 1. For the results corresponding to larger values of K, we can see a slight improvement in the performance of WSSR+ in the setting of both active learning and constrained clustering. The improvement brought by active learning is much larger, which can be observed on the performance of both PLCC and CSP as well. This demonstrates the advantage of incorporating informative points as compared to points that are randomly sampled.

4.8 Conclusions & Future Work

In this work, we first propose a sparse subspace clustering method that constructs the data affinity matrix by expressing each point as a linear combination of a subset of other points. The sparse coefficients can be found by solving a convex optimisation problem that has an elastic net objective and simplex constraint. We term the full optimisation problem as *Weighted Sparse Simplex Representation (WSSR)*. We show that one sub-problem of WSSR can be solved analytically via setting a system of linear equations. We also detail two possible approaches for solving the full problem: one approach is to solve it through a standard quadratic programming solver; another is through projected gradient descent.

In the presence of available side information, we also introduce a constrained clustering version termed WSSR+, that achieves effective performance improvement. In the scenario where the class information can be queried sequentially over time, we demonstrate how WSSR+ can fit smoothly into an active learning framework to query the points that can be the most beneficial for cluster performance improvement. Experiments are conducted on both synthetic data and real data to demonstrate the effectiveness of our proposed methodology in the following three scenarios: (a) subspace clustering without any side information; (b) constrained subspace clustering with a fixed amount of side information; and (c) constrained subspace clustering in an active learning framework in which the class labels can be queried sequentially over time.

We outline a few directions for future research. Firstly, we would like to strengthen our proposed methodology by exploring more principled approaches for choosing the neighbourhood size. Secondly, the absolute cosine similarity is only a natural affinity measure for points lying in linear subspaces. It would be interesting to explore other affinity measures that can potentially be used for data from either linear or affine subspaces. Thirdly, we model each point as a linear combination of other points that lie in its local neighbourhood. This idea is also utilised in manifold clustering setting, thus it would be interesting to compare the performance of our method with other manifold clustering methods when the data exhibit manifold structure. Finally, it is worth pointing out that what we propose is a general framework that can potentially be embedded in other settings, for example deep learning (Goodfellow et al., 2016), where data are abundant in applications such as image analysis and text mining. Exploring the possibility of either applying our proposed method to learned deep features with Convolutional Neural Networks (CNN) (LeCun et al., 1998) or embedding our proposed WSSR self-expressiveness model into an Antoencoder (AE) (Goodfellow et al., 2016) architecture would be yet another interesting direction for future research.

4.A Appendix

4.A.1 WSSR+ Experiments on UCI Benchmark Data

In this section, we conduct further experiments to compare WSSR+ with other state-of-the-art constrained clustering methods on data sets that do not exhibit subspace structure. The experiments are conducted on four UCI benchmark data sets (Dua and Graff, 2017), which have been used previously to demonstrate the effectiveness of constrained spectral clustering methods (Wang et al., 2014; Liu et al., 2018). A summary of the data characteristics can be found in Table 4.9.

Data sets	No. of points (N)	No. of features (P)	No. of clusters (K)
iris	150	4	3
wine	178	13	3
ecoli	336	343	8
glass	214	9	6

Table 4.9: A summary of the UCI benchmark data sets.

Performance results in terms of clustering accuracy are reported in Table 4.10 under varying proportions of side information. For each proportion $p\%$ of side information, we run all experiments for 20 trials and report the median clustering accuracy along with the standard deviation. For each data set, the best performance results are highlighted in bold, and the second best performance results are underlined.

It is worth noting that all the initial clustering accuracy scores achieved by WSSR here are significantly better than that produced by spectral clustering as reported in Liu et al. (2018). As a result, the performance results for all three competing methods are much better than what have been previously reported. With that said, it can be seen that extra side information can have a negative impact on the resulting accuracy. For example, this is the case for PLCC on both *ecoli* and *glass* data sets, in which the performance degrades with the increase of side information.

Overall, WSSR+ has a favourable performance against all three competing methods. In particular, it enjoys the best performance across all side information levels on the *wine* data set and remains one of the top two performers on the remaining data sets. As

Data	Pct.	WSSR	WSSR+		PLCC		CSP		LCVQE	
			Med	Std	Med	Std	Med	Std	Med	Std
iris	10%	0.97	**0.97**	0.00	**0.97**	0.00	$\underline{0.94}$	0.14	0.90	0.01
	20%		$\underline{0.97}$	0.00	**0.98**	0.01	**0.98**	0.07	0.92	0.02
	30%		$\underline{0.98}$	0.01	**0.99**	0.01	$\underline{0.98}$	0.12	0.93	0.01
wine	10%	0.83	**0.86**	0.02	$\underline{0.85}$	0.01	0.68	0.06	0.71	0.02
	20%		**0.88**	0.02	$\underline{0.86}$	0.01	0.75	0.07	0.73	0.03
	30%		**0.88**	0.02	$\underline{0.87}$	0.04	0.85	0.05	0.71	0.04
ecoli	10%	0.78	**0.77**	0.01	0.67	0.06	$\underline{0.76}$	0.03	**0.77**	0.03
	20%		$\underline{0.80}$	0.01	**0.82**	0.02	0.76	0.02	$\underline{0.80}$	0.05
	30%		**0.81**	0.02	0.71	0.05	0.76	0.02	$\underline{0.80}$	0.03
glass	10%	0.68	**0.69**	0.01	$\underline{0.67}$	0.04	0.65	0.09	0.58	0.02
	20%		**0.69**	0.01	$\underline{0.66}$	0.01	**0.69**	0.04	0.60	0.04
	30%		$\underline{0.70}$	0.01	0.60	0.03	**0.72**	0.04	0.61	0.04

Table 4.10: Clustering accuracy of various spectral-based constrained clustering methods on UCI benchmark data sets.

opposed to the adverse effects that have been observed in the competing methods, the performance of WSSR+ improves consistently with the increase of side information. The performance of PLCC appears to be similarly competitive to that of WSSR+. However the standard deviation from PLCC can be comparatively big on data sets such as *ecoli* and *glass*. The performance variability in PLCC undermines its reliability and seemingly high median clustering accuracy.

To further investigate the stability of various methods, we provide detailed performance visualisations for all methods on all data sets in Figure 4.A.1. Each plot presents the minimum, median, and maximum performance of each constrained clustering method across 20 trials. The proportion of known class labels $p\%$ range from 0.1 to 1.0, with 1.0 being all class labels are known. High variability can be observed from the performance of CSP (on *iris* data set) and LCVQE (on *wine* and *glass* data sets). It also becomes obvious that even for methods that have relatively small variability, such as PLCC and WSSR+, consistent performance improvement is not always achieved on all data sets. In particular, the clustering accuracy of PLCC decreased when the available side information increased to 40% on the iris data. Taking into account both consistent and stable performance improvement, WSSR+ is competitive against these state-of-the-art methods.

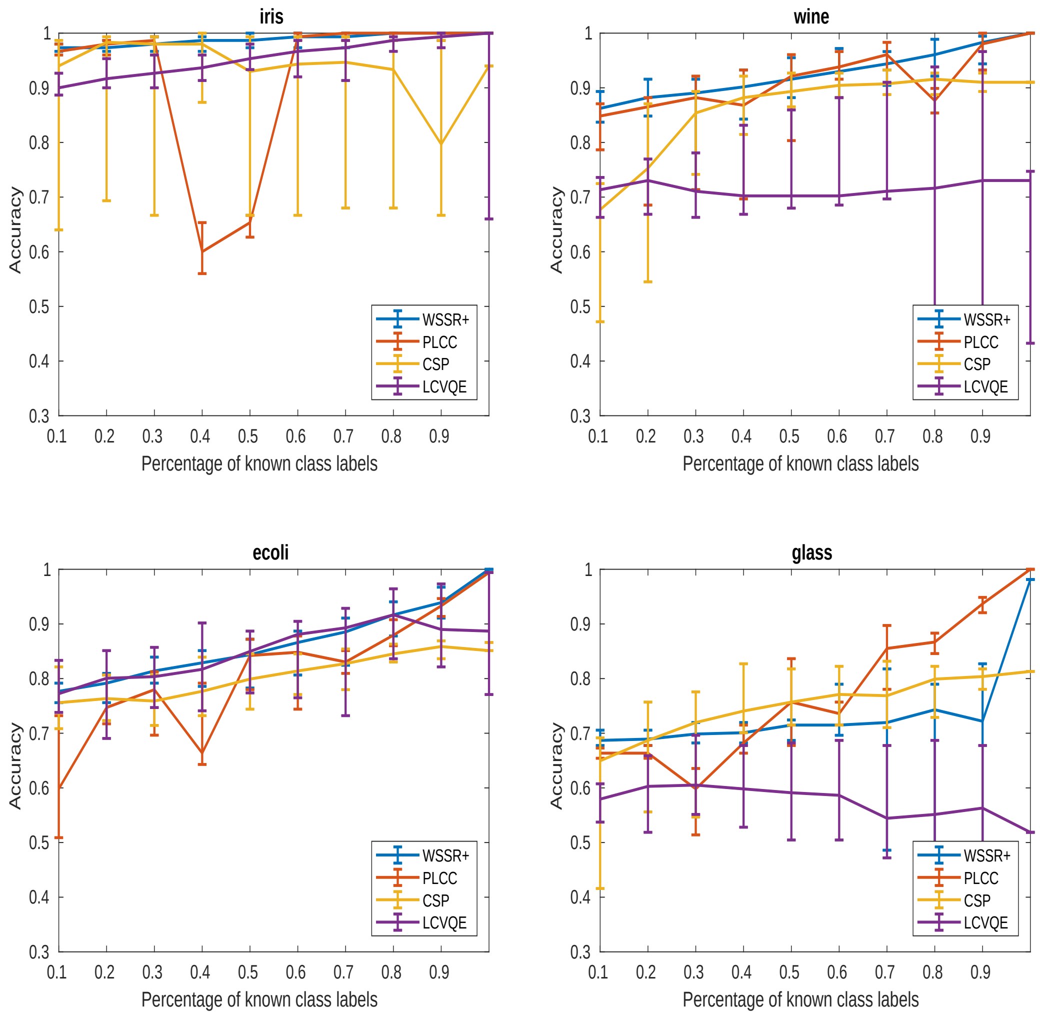


Figure 4.A.1: The clustering accuracy (min, median, max) of various constrained clustering algorithms over 20 trials.

4.A.2 WSSR+ Experiments on Cancer Gene Data

In the previous section, we have demonstrated the effectiveness of WSSR+ in the setting of constrained clustering on generic benchmark data that do not necessarily have subspace structure. Next we inspect the performance of WSSR+ in an active learning framework, where the additional side information are being acquired sequentially over time to enable effective performance improvement. Experiments are conducted on gene expression data, which have been shown to enjoy subspace structure (McWilliams and Montana, 2014). Each subtype of gene expression data forms a subspace of its own. The following three data sets have been previously experimented with (Li et al., 2017, 2018b) to demonstrate the effectiveness of subspace clustering: *St. Jude Leukemia*, *Lung Cancer*, and *Novartis*

BPLC. A summary of the basic data characteristics can be found in Table 4.11.

Data sets	No. of points (N)	No. of features (P)	No. of clusters (K)
St. Jude Leukemia	248	985	6
Lung Cancer	197	1000	4
Novartis BPLC	103	1000	4

Table 4.11: Summary information on the gene expression data sets.

When there is no side information available, we compare the cluster performance of WSSR with state-of-the-art subspace clustering methods SSC (Elhamifar and Vidal, 2013) and S3C (Li and Vidal, 2015). We have briefly introduced SSC in Section 4.6, and S3C is a generalisation of SSC that simultaneously solves for the affinity matrix and the cluster assignment. In addition, we compare the performance of WSSR+ with varying proportion of side information against that of CS3C (Li et al., 2017) and CS3C+ (Li et al., 2018b). Both of these constrained methods have shown excellent performance on the three gene expression data sets as compared to several other subspace methods. For all competing algorithms, we use the active strategy proposed in Peng and Pavlidis (2019) to query points based on the current cluster assignment.

The parameters for all SSC-based methods are tuned to each data set to produce the best results, as reported in Li et al. (2017). For WSSR, we use $\rho = 0.01$ and tune the number of nearest neighbours k for each data set. We use the hard version of S3C, as we are certain about the validity of the side information. The performance results for all methods with varying levels of side information are reported in Table 4.12.

When comparing the performance without side information, we observe that WSSR yields the best accuracy in two out of three cases. Although both S3C and WSSR achieve the same level of accuracy on the Leukemia data set. It is worth noting that WSSR achieves perfect clustering accuracy on the Novartis data, whereas the two constrained methods CS3C and CS3C struggle to make any improvement with up to 30% of side information. On the other two data examples, it can be seen that WSSR+ is able to make effective performance improvement with the increase of available side information. Both CS3C and CS3C+ are able to make some improvement on the Cancer data, but struggle

Data	Pct.	WSSR	WSSR+	SSC	S3C	CS3C	CS3C+
	10%		**0.98**			**0.98**	**0.98**
Leukemia	20%	**0.98**	**0.98**	0.97	**0.98**	0.98	0.98
	30%		**0.99**			0.98	0.98
	10%		0.94			**0.97**	0.96
Cancer	20%	0.92	**0.98**	0.95	**0.96**	0.97	0.97
	30%		**1.00**			0.99	0.99
	10%		**1.00**			0.98	0.98
Novartis	20%	**1.00**	**1.00**	0.97	0.97	0.98	0.98
	30%		**1.00**			0.98	0.98

Table 4.12: Performance comparison (using accuracy) with state-of-the-art (constrained) subspace clustering methods.

to benefit from the additional side information on the other two data sets.

The percentage of side information here refers to the percentage of points whose class memberships are known to us. This is different from the notion as used in Li et al. (2017) and Li et al. (2018b). In the experiments therein, 10% side information would mean that 10% of the entries in the side information matrix Ψ are known. Whereas in our case, 10% of side information would translate into only about 1% of filled entries of the side information matrix Ψ. This makes the performance of WSSR+ more notable given how little additional information it requires to reach perfect clustering accuracy.

Chapter 5

Clustering the Amazon Web-Scraped Text Data

The success of various types of subspace clustering methods have been demonstrated through a wide range of applications from motion segmentation to image recognition. What these applications have in common is that the feature representation for both the image data and the video (image frames) data are high-dimensional. Nevertheless, the movement of an object captured through different images / videos can be well summarised in a much lower dimensional subspace. This motivated us to explore the potential of applying subspace clustering methods to other data types that also enjoy such subspace properties. The problem of clustering text data has been studied extensively in the document clustering literature. However less attention has been paid to addressing the problem within the strand of subspace clustering techniques.

In this chapter, we first study and apply a number of clustering techniques on a US Amazon web-scraped data set with product names. We show that the data set represented using the TF-IDF matrix enjoys subspace structure as a whole and within each product category. Finally, we propose a simple subspace clustering algorithm that relies on principal angles to cluster the data. Experimental results on identifying product categories from product names indicate that the algorithm can be competitive against state-of-the-art clustering algorithms.

5.1 Introduction

In many countries, the inflation and deflation rates as reflected by the price indices indicate the general service and product price fluctuations in the market, and serve as indicators for the national economic dynamics. Consumer Price Index (CPI) is the official price index in the UK, which is published on a regular basis by the Office for National Statistics (ONS). Traditionally, national price indices are published based on a basket of product prices that are manually collected. Such a data collection procedure is not only time consuming but also limited in the amount of data that can possibly be obtained. Since early 2014, the ONS set up the Big Data Project to investigate the benefits and challenges of using novel data sources such as web-scraped data to improve the current price index generating procedure.

Web-scraped data are more advantageous than traditionally collected data from the perspective that they can be easily scraped in a huge amount and at a high frequency with a low monetary cost. However, the use of web-scraped data in the index generating procedure is not straightforward. First, the size and frequency of the web-scraped data require large computational power for processing. Secondly, the categorisation on many retail websites from which the data are scraped do not necessarily match the categorisation adopted by the ONS for index generation purposes. Thirdly, product information (names, prices, etc.) might not be correctly scraped due to the quality of the web-scrapers and the occasional out-of-stock products.

Some of these challenges can be suitably tackled via clustering techniques. For example, in order to map the website product categorisation into the categorisation used by the ONS for price index generation, it is important to identify the main categories of the web-scraped data. This task fits naturally into the goal of clustering, which is to identify the main groups in a data set such that the data points in the same group are more similar to each other compared to data points in different groups.

The main types of information that come with the web-scraped data are the product names and the price information. However, it would not be reasonable to conduct clustering based on the price information. This is because products from different categories can have similar prices. Instead, a human would be able to identify the general category of a product based on the semantic meaning of the title.

The fact that most product names are composed of short texts has two important implications. Firstly, each word within each product name is likely to appear only once. Secondly, the vast majority of pairs of product names have no words in common. These properties of the data pose challenges to established text mining algorithms. It also makes it difficult for statistical methods that employ generative models such as Latent Dirichlet Allocation (Blei et al., 2003), which requires long document length to achieve reliable parameter estimates.

If one uses the standard Term Frequency–Inverse Document Frequency (TF-IDF) representation (Salton and Buckley, 1988), the resulting data matrix for such document collections would be both sparse and high-dimensional. In this setting, it is sensible to argue that texts that share even a few number of words are very similar to each other. Therefore, associating clusters with linear combinations of the features (i.e. linear subspaces) is reasonable. We propose a simple subspace clustering algorithm called **Minimum Angle Clustering (MAC)** that is motivated by the characteristics of short texts. It first identifies a large number of subspaces that contain few but very similar observations. Then an appropriate dissimilarity measure is used to merge these subspaces into meaningful clusters.

The rest of this chapter is organised as follows. In Section 5.2, we first provide a description for the US Amazon web-scraped data that we will be using throughout the rest of this chapter. We then introduce two document processing techniques that transform text data into vector space representations. In Section 5.3, we review document clustering approaches that we later apply to the US Amazon web-scraped data. A simple subspace clustering algorithm is proposed in Section 5.4, which exhibits favourable performance on the data as compared to other competing algorithms as shown in the experimental results in Section 5.5. Finally, we conclude the chapter in Section 5.6 and discuss a few routes for future research.

5.2 Vector Space Representation

The data that we use in this chapter is a publicly available data set that contains the US Amazon web-scraped electronic product information. It is obtained from the Billion Prices Project (BPP) website, which is an academic initiative founded in 2008 that uses web-scraped prices for economic research (Cavallo, 2017). They use their own web-scrapers to collect data from hundreds of online retailers around the world on a daily basis. The Amazon web-scraped data set contains the following features:

- Date: the date on which the product information is scraped;
- Product name: the name of the product that is shown on Amazon website;
- Product price: the listed price of the product;
- Product URL: the website link to the product;
- Product category: the assigned category label for the product.

A summary of the product category information is shown in Table 5.1.

Electronics	**Home and appliances**	**Mix**	**Office products**	**Pharmacy and health**
804	389	1490	111	127

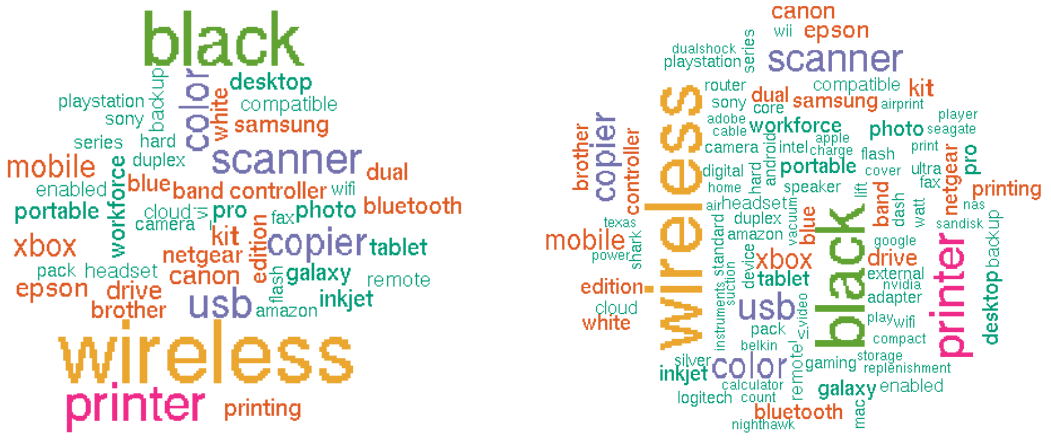
Table 5.1: A summary of the five categories in the Amazon web-scraped data.

From now on, we will refer to the Amazon web-scraped data as the Amazon data. In order to develop a machine-level understanding for language modelling tasks, it is important to first represent the text data numerically. Vector space representations are algebraic models for representing text documents as vectors, and they serve as the building blocks for various language models. In this section, we introduce the reader to two types of representations in which each dimension in the vector space representation corresponds to a unique term in the text.

5.2.1 Document Term Matrix (DTM)

The most straightforward way to transform text data into vector space representation is through the *Document Term Matrix (DTM)* (Larson, 2010). It is a word count matrix that describes for every document the number of times that each word has occurred in a specific document. The number of rows in the matrix corresponds to the number of documents N, and the number of columns corresponds to the number of unique terms P. The DTM representation X is an $N \times P$ matrix, in which x_{ij} represents the number of times term j ($j \in \{1, \ldots, P\}$) has appeared in document i ($i \in \{1, \ldots, N\}$).

DTM representation does not take into account the fact that documents may have different lengths. As a result, certain words may appear more often in some documents than others because the lengths those documents are relatively long. In addition, DTM representation does not take into account the possibility that some common words may appear frequently across many documents. Such words may carry less information than those words that appear less often but only in a selected few documents. This can be shown in the two word clouds in Figure 5.2.1, which contains the top-50 and top-100 most frequent words in the Amazon data. It can be seen that words such as ‘black’ and ‘color’ are among the top-50 most frequent words, but they could potentially occur in multiple documents from different categories.



(a) Top-50 most frequent words.

(b) Top-100 most frequent words.

Figure 5.2.1: Word clouds containing the most frequent words in the Amazon data.

To address these issues, various weighting schemes have been proposed in the literature to improve upon DTM. One of the most commonly used weighting scheme is the *Term Frequency - Inverse Document Frequency (TF-IDF)* (Larson, 2010), which we will introduce in the next section.

5.2.2 Term Frequency - Inverse Document Frequency (TF-IDF)

In this section, we introduce the Term Frequency - Inverse Document Frequency (TF-IDF) weighting scheme (Salton and Buckley, 1988), which incorporates two weight measures that address the aforementioned issues with the DTM representation. The first measure is *term frequency* – it is the proportion of the number of times that word j appears in document i, x_{ij}, out of the total number of words in document i. The term frequency is defined as

$$\text{TF}_{ij} = \frac{x_{ij}}{\sum_{j=1}^{P} x_{ij}}. \tag{5.2.1}$$

The second measure is *inverse document frequency*, which measures how often a term appears across all documents. The inverse document frequency is defined as

$$\text{IDF}_j = \log\left(\frac{N}{\sum_{i=1}^{N} \mathbb{1}_{\{x_{ij}>0\}}}\right), \tag{5.2.2}$$

in which $\mathbb{1}_{\{x_{ij}>0\}}$ is an indicator function defined as:

$$\mathbb{1}_{\{x_{ij}>0\}} = \begin{cases} 1, & x_{ij} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{5.2.3}$$

According to the IDF measure, the more documents that a word appears in, the less important it is. By combining the TF and IDF measures together, the TF-IDF representation is obtained as

$$\text{TF-IDF}_{ij} = \text{TF}_{ij} \times \text{IDF}_j,\ \forall\, i, j. \tag{5.2.4}$$

In practice, the TF-IDF representation vectors are often normalised to have unit ℓ_2-norm so as to retain only the direction of the document vectors (Dhillon and Modha, 2001).

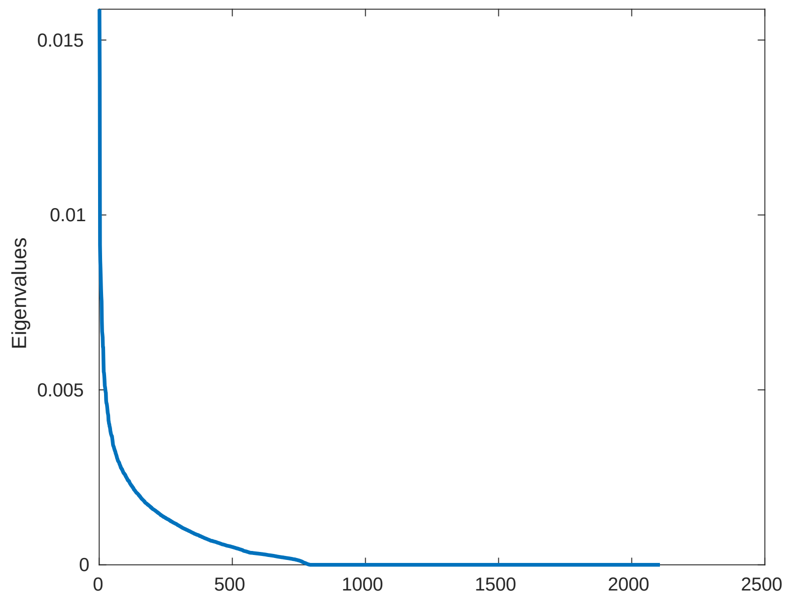
The resulting TF-IDF representation for the US Amazon data contains 2921 rows (product names) and 2106 columns (unique terms). For each product category, a summary of the 10 most highly weighted words in the TF-IDF matrix is shown in Table 5.2. It is clear to see that many of these words are indeed indicative of the category that they come from. For example, one can easily tell that ‘printer’, ’usb’, and ’xbox’ are terms from the ‘Electronics’ category; ‘colgate’, ‘toothpaste’, and ‘toothbrush’ are clearly terms from ‘Pharmacy and Health’ category.

Electronics	*Home and Appliances*	*Mix*	*Office Products*	*Pharmacy and Health*
black	loreal	amp	file	oz
wireless	cream	dewalt	folders	colgate
printer	lock	sander	hanging	white
allinone	shine	corded	pendaflex	toothpaste
white	hansen	hitachi	positions	coffee
usb	sally	orbital	reinforced	maker
epson	coppertop	volt	surehook	pk
one	master	saw	size	count
xbox	duracell	cc	letter	toothbrush
canon	assorted	speed	blendngo	whitening

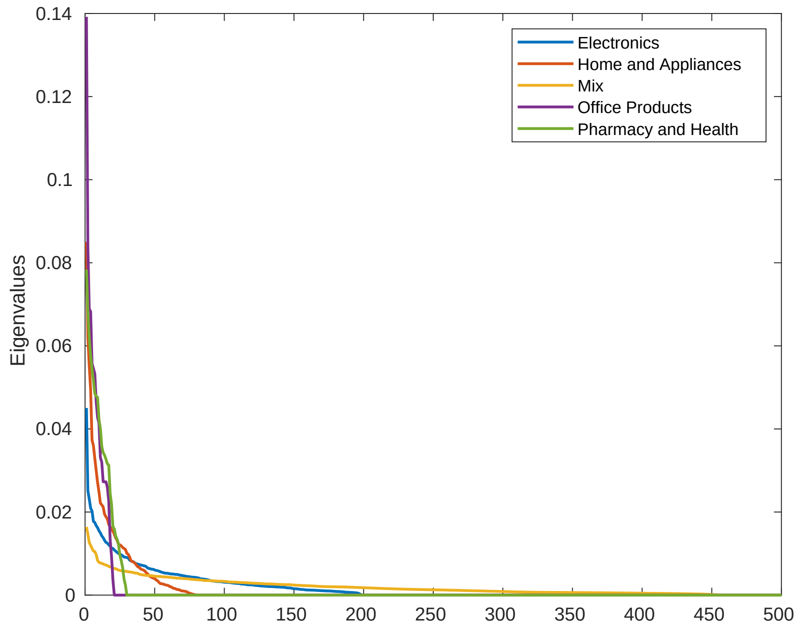
Table 5.2: A list of the ten most highly weighted words within each product category according to the TF-IDF representation.

We further explore the structure of the TF-IDF matrix as a whole and within each category by inspecting the corresponding eigenvalues. Figure 5.2.2 (a) shows the eigenvalues of the full TF-IDF matrix in descending order. It can be seen that less than half of the 2106 eigenvalues are greater than zero. This indicates that the variability of the data as a whole can be captured within a much lower dimensional subspace.

In addition, we divide the full TF-IDF matrix into five sub-matrices with each sub-matrix corresponding to one of the five product categories. In Figure 5.2.2 (b), we plot the eigenvalues for each sub-matrix in descending order. The upper limit of the x-axis is 500, as the number of eigenvalues that are significantly different from zero is far smaller than 500 for all categories. This observation confirms that the TF-IDF representation for each product category lies in a much lower dimensional subspace as compared to the original dimension.



(a) The eigenvalues of the full TF-IDF matrix (in descending order).



(b) The eigenvalues of the TF-IDF matrix for each product category (in descending order).

Figure 5.2.2: An illustration of the subspace structure of the US Amazon data.

5.3 Document Clustering Methods

In this section, we review two document clustering techniques: Principal Direction Divisive Partitioning (PDDP) (Boley, 1998), and Latent Dirichlet Allocation (LDA) (Blei et al., 2003). PDDP is a top-down hierarchical document clustering technique that treats each document as a vector in the Euclidean space. LDA is a probabilistic topic modelling technique that assigns labels to words, and the document labels are obtained using the dominant word label in the document.

5.3.1 Principal Direction Divisive Partitioning (PDDP)

Principal Direction Divisive Partitioning (PDDP) (Boley, 1998) is a top-down hierarchical document clustering technique. PDDP considers the data to be a collection of text documents $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$. Each text document is first transformed into a vector space representation $\boldsymbol{x}_i \in \mathbb{R}^P$, where P denotes the dimension of the vector space representations for the text documents, i.e. the size of the vocabulary. Either of the vector space representations previously discussed in Section 5.2 can be used here.

As an initial step, the algorithm divides the whole data set into two groups. Then each of these two groups will be iteratively divided into smaller groups. This process continues progressively, which results in a hierarchical tree representation of the data. The main questions to be addressed within this divisive hierarchical clustering framework include the following:

(1) How to split a group of data points into two?

(2) Which group among the existing groups should be split first?

(3) When should the process terminate?

We will address each of these questions in the rest of this section.

As is reflected in the name Principal Direction Divisive Partitioning, the algorithm uses the first principal component direction to split a group of points. Take the initial split of the full data matrix $X \in \mathbb{R}^{N\times P}$ as an example, the principal components of the data can be obtained from the eigen-decomposition of its covariance matrix S, which can be calculated as

$$S = \left(X - \mathbf{1}\boldsymbol{\mu}^{\mathsf{T}}\right)^{\mathsf{T}} \left(X - \mathbf{1}\boldsymbol{\mu}^{\mathsf{T}}\right), \tag{5.3.1}$$

where $\boldsymbol{\mu} = \frac{1}{N} X^{\mathsf{T}}\mathbf{1}$ is the feature-wise mean vector. The eigen-decomposition can be expressed as $SV = \Lambda V$, in which the columns of $V = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_P]$ are the principal component vectors, and the diagonal entries of Λ correspond to the eigenvalues.

PDDP splits the data X into two groups according to the projected values along the first principal component direction as follows

$$\left(X - \mathbf{1}\boldsymbol{\mu}^{\mathsf{T}}\right) \boldsymbol{v}_1. \tag{5.3.2}$$

It can be shown that Eq. (5.3.2) can equally be expressed in terms of the Singular Value Decomposition (SVD) of the data, $X = U\Sigma V^{\mathsf{T}}$, as

$$\left(X - \mathbf{1}\boldsymbol{\mu}^{\mathsf{T}}\right) \boldsymbol{v}_1 = \sigma_1 \boldsymbol{u}_1, \tag{5.3.3}$$

in which σ_1 is the largest singular value of X, and $\boldsymbol{u}_1 \in \mathbb{R}^N$ is the first column vector in the left singular matrix U. The *Lanczos algorithm* can be used for efficient computation of $\sigma_1 \boldsymbol{u}_1$, which takes advantage of the sparsity in the data (Golub and Van Loan, 2013).

The division of the points into two groups relies on the sign of Eq. (5.3.2). That is, entries in Eq. (5.3.2) with positive values are assigned to one group and entries with negative values are assigned to another group. Entries with zero values are assigned randomly into either of the two groups. In order to decide which group to split first, a cluster quality measure is required. For a sub-matrix X_k that contains a subset of the points in X, a measure of cluster cohesiveness is defined in Boley (1998) as

$$\|X_k - \mathbf{1}\boldsymbol{\mu}_k^{\mathsf{T}}\|_F, \tag{5.3.4}$$

where $\boldsymbol{\mu}_k$ is the feature-wise mean vector for X_k. This measure reflects the overall distance between each document vector to its cluster centre. The algorithm selects the group with the highest value when evaluated with Eq. (5.3.4), which is the least cohesive among all groups. The algorithm terminates either when a maximum number of splits is reached, or when all existing groups reached a certain level of cohesiveness.

The main advantage of PDDP lies in its efficiency in handling high-dimensional data. However, the criterion of splitting the data along the first principal direction by the sign of the projected points is not always optimal, as it is possible that a sub-group of points span both the negative and positive regions along the first principal direction. Other splitting criteria have been proposed in the literature, for example, the variant iPDDP (Tasoulis and Tasoulis, 2008) splits the data on the first projection at where the largest gap is. However, it does not seem to perform very well when the noise level is high. Density-based PDDP (dePDDP) (Tasoulis et al., 2010) splits the cluster by the lowest density point.

5.3.2 Latent Dirichlet Allocation (LDA)

Topic modelling algorithms use statistical methods to discover the thematic structure that pervades a large unstructured collection of text documents (Blei, 2012). Latent Dirichlet Allocation (LDA) is a probabilistic topic modelling technique which assumes that an underlying generative process is accountable for producing the words that are observed in the text documents (Blei et al., 2003). The generative process is governed by a joint probability distribution which includes both observed and hidden variables. Given the generative process, the words in a document are generated in the following steps:

(1) Randomly choose a distribution over topics (per document),

(2) Randomly choose a topic from the distribution of topics (per word),

(3) Randomly choose a word from the distribution over words.

This process reflects the fact that each document contains multiple topics with different proportions. The main task of LDA is to use the observed variables (text documents) to infer the hidden variables that are responsible for the topic structure. The problem of inferring the hidden variables is the problem of computing its posterior distribution given the observed variables. Next, we provide a more formal description of LDA using the notations summarised in Table 5.3.

Symbol	**Meaning**
N	The total number of documents.
K	The total number of topics.
k	The index for the k-th topic ($k \in \{1, \dots, K\}$).
V	The size of the vocabulary, i.e. the total number of unique terms in the text documents.
V_i	The number of terms in the i-th document.
$\boldsymbol{w}_i \in \mathbb{R}^V$	The observed words in the i-th document, in which w_{ij} denotes the presence of the j-th term in the i-th document ($i \in \{1, \dots, N\}$ and $j \in \{1, \dots, V\}$).
$\boldsymbol{z}_i \in \mathbb{R}^V$	The topic labels for the i-th document, in which z_{ij} denotes the topic of the j-th term in the i-th document ($i \in \{1, \dots, N\}$, $j \in \{1, \dots, V\}$).
$\boldsymbol{\theta}_i \in \mathbb{R}^K$	The topic proportions for the i-th document, in which θ_{ik} denotes the proportion of words that belong to topic k in document i. We have that $\|\boldsymbol{\theta}_i\|_1 = \sum_{k=1}^K \theta_{ik} = 1$.
$\boldsymbol{\phi}_k \in \mathbb{R}^V$	The probabilities of drawing words from a given topic k, in which ϕ_{kj} denotes the probability of drawing the j-th word in the vocabulary from topic k.

Table 5.3: A summary of notations for LDA.

Dirichlet distribution is suitable for modelling prior beliefs on more than two proportions. Its probability density function for the distribution over topics of an arbitrary document can be written as follows

$$\mathbb{P}(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}, \tag{5.3.5}$$

where $B(\boldsymbol{\alpha})$ is the multivariate Beta function

$$B(\boldsymbol{\alpha}) = \frac{\prod_{k=1}^K \alpha_k}{\Gamma\left(\sum_{k=1}^K \alpha_k\right)}. \tag{5.3.6}$$

The LDA model assumes the Dirichlet distribution for both the *distribution over topics* Dir$(\boldsymbol{\theta}|\boldsymbol{\alpha})$, and the *distribution over words* Dir$(\boldsymbol{\phi}|\boldsymbol{\beta})$ under each topic. In Eq. (5.3.5), $\boldsymbol{\theta} = [\theta_1, \dots, \theta_K]^\mathsf{T}$ denotes the vector of topic proportions, and $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_K]^\mathsf{T}$ is the vector of *concentration parameters* that reflect the prior belief on the distribution over topics. For the distribution over words $\mathrm{Dir}(\boldsymbol{\phi}|\boldsymbol{\beta})$, $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_V]^\mathsf{T}$ denote the vector of probabilities for sampling words from the topic, and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_V]^\mathsf{T}$ reflects the prior belief on the distribution over words. Note that same word could potentially be drawn from more than one topic, but the probability of the same word being drawn from different topics is dependent on the specific topic.

The generative process first determines the distribution over topics in a document. Once the topic distribution is determined, words will then be generated from each topic accordingly. The probability of drawing the j-th word from the i-th document, w_{ij}, from any of the K topics can be calculated as

$$\mathbb{P}(w_{ij}) = \sum_{k=1}^{K} \mathbb{P}(w_{ij}|z_{ij} = k)\mathbb{P}(z_{ij} = k), \tag{5.3.7}$$

in which $\mathbb{P}(w_{ij}|z_{ij} = k)$ denotes the conditional probability of drawing word w_{ij} given that it is from topic k, and $\mathbb{P}(z_{ij} = k)$ is the probability that the word w_{ij} is drawn from topic k. Practically, it is given by the proportion of words that are assigned to topic k in document i divided by the total number of words in the document. It is worth noting that this is different from the proportion of words that belong to topic k divided by the total number of distinct words, as the same word can appear in multiple topics.

Combining the probabilities for obtaining all words in all documents, we obtain the joint probability distribution for the generative process as follows (Blei et al., 2003)

$$\mathbb{P}(\boldsymbol{w}; \boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{i=1}^{N} \mathbb{P}(\boldsymbol{\theta}_i; \boldsymbol{\alpha}) \prod_{k=1}^{K} \mathbb{P}(\boldsymbol{\phi}_k; \boldsymbol{\beta}) \prod_{j=1}^{V_i} \mathbb{P}(z_{ij}|\boldsymbol{\theta}_i)\mathbb{P}(w_{ij}|\boldsymbol{\phi}_{z_{ij}}). \tag{5.3.8}$$

For each document i ($i \in \{1, \ldots, N\}$), $\mathbb{P}(\boldsymbol{\theta}_i; \boldsymbol{\alpha})$ gives the document-topic distribution. For each topic k ($k \in \{1, \ldots, K\}$), $\mathbb{P}(\boldsymbol{\phi}_k; \boldsymbol{\beta})$ gives the topic-word distribution. For each word j ($j \in \{1, \ldots, V_i\}$) in document i, first a topic z_{ij} is chosen according to $\mathbb{P}(z_{ij}|\boldsymbol{\theta}_i)$, then a word w_{ij} is chosen within topic z_{ij} with probability $\mathbb{P}(w_{ij}|\boldsymbol{\phi}_{z_{ij}})$. The goal is to infer the topic labels of all words such that Eq. (5.3.8) is maximised.

The actual model inference procedure for the model parameters can be done in various ways. Following a Frequentist approach, the model parameters can be obtained via the *Expectation-Maximisation (EM)* procedure. Following a Bayesian approach, *Gibbs Sampling (GS)* can be used to estimate the parameter values. The final label of each document can thus be determined by the majority label out of all topic labels for all words in the document.

5.4 Minimum Angle Clustering (MAC)

In this section, we propose a simple algebraic subspace clustering technique called Minimum Angle Clustering (MAC) (Peng et al., 2018), which is motivated by the problem of clustering short product names in the Amazon data set. Given N product names, the goal is to identify a pre-specified K main product categories. The algorithm first merges very similar points (product names) together to form N_c groups / subspaces, where $N_c > K$. Then, a subspace dissimilarity measure is used to merge these subspaces into K meaningful clusters.

We first transform the product names into vector space representations using the Term Frequency–Inverse Document Frequency (TF-IDF) representation, which is previously introduced in Section 5.2.2. Since each product name is very short, and different product names contain different words, the TF-IDF representation is both sparse and high-dimensional. We denote the TF-IDF matrix as $X \in \mathbb{R}^{N \times P}$, in which the rows correspond to the product names and the columns correspond to the unique words.

Subspace clustering assumes the set of data points $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$ are drawn from a union of K subspaces. Each point $\boldsymbol{x}_i \in \mathcal{X}$ is assumed to lie on (or close to) a relatively low-dimensional subspace. Recall from Chapter 1 Section 1.2, a q_k-dimensional linear subspace $\mathcal{S}_k$ is defined as

$$\mathcal{S}_k = \left\{\boldsymbol{x}_i \in \mathbb{R}^P : \boldsymbol{x}_i = V_k \boldsymbol{y}_i\right\}, \tag{5.4.1}$$

where $V_k \in \mathbb{R}^{P \times q_k}$ is an orthonormal matrix defining the basis for subspace $\mathcal{S}_k$, and $\boldsymbol{y}_i \in \mathbb{R}^{q_k}$ is the representation of $\boldsymbol{x}_i$ in terms of the column vectors of V_k. The goal of subspace clustering is to both identify the K subspaces, and to identify the subspace allocations for all points.

In the context of our problem, features of the TF-IDF representation correspond to the unique words. It is therefore reasonable to assume that texts that share a combination of words are similar to each other. Reduced row echelon form (RREF) can provide useful grouping information based on linear combinations of the features (Gear, 1994, 1998). We thus propose to first transform the transpose of the TF-IDF matrix $X^{\top}$ into its reduced row echelon form, which can be achieved by applying the well-known Gauss-Jordan elimination (Golub and Van Loan, 2013). In this process, a sequence of row operations are performed to bring $X^{\top}$ into a form that satisfies [1]:

1. the leftmost non-zero entry of each row is 1;
2. the leftmost non-zero entry of each row is the only non-zero entry in the corresponding column;
3. for any two different leftmost non-zero entries, one located in row i, column j and the other in row s, column t: if $s > i$, then $t > j$;
4. rows in which every entry is zero are beneath all rows with non-zero entries.

Let F denote the reduced row echelon form of $X^{\top}$. The columns of F that have a single non-zero element are called *pivot columns*. The first column of F is always a pivot column. Moreover, column j $(j > 1)$ is a pivot column if and only if the j-th column of $X^{\top}$ (i.e. $\boldsymbol{x}_j$) cannot be expressed as a linear combination of the previous columns (i.e. columns 1 to $(j-1)$). If the j-th column is a non-pivot column of F, then the non-zero elements in this column specify the coefficients of the linear combination of the previous pivot columns that yield the j-th column vector.

Since points that can be written as linear combinations of each other belong to the same linear subspace, F provides valuable information to identify clusters spanning different subspaces. A simple approach to identify subsets of points that belong to the same linear subspace through F is the following. Define the matrix $Y \in \{0,1\}^{P \times N}$

[1]See Golub and Van Loan (2013) for numerically stable algorithms to perform this operation.

in which $Y_{i,j} = \mathbb{1}_{\{F_{i,j} \neq 0\}}$, where $\mathbb{1}_{\{\cdot\}}$ is the indicator function that returns one if its argument is true and zero otherwise. That is, $Y_{i,j}$ indicates whether a point $\boldsymbol{x}_j$ is in the linear combination to approximate $\boldsymbol{x}_i$, or the other way around. As such, the adjacency matrix, $A = Y^{\mathsf{T}}Y$, provides pairwise connectivity information between points which can then be used to obtain a graph G whose N_c connected components are subsets of points that can be expressed as linear combinations of each other.

For the problem of clustering very short texts, the graph G has a very large number of connected components, many of which contain only a single point. Texts that belong to the same connected component are very similar, hence this partitioning is very accurate in terms of purity. However, it is of no practical use because it fails to capture the main groups. Figure 5.4.1 provides the histogram of the number of points in each connected component of G for the Amazon data set. As the figure shows, the vast majority of connected components contain less than ten points, while the mode of this distribution is at one.

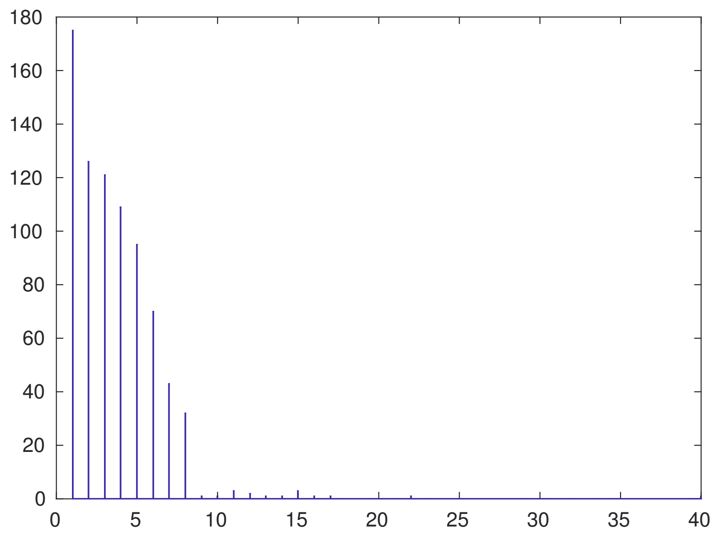


Figure 5.4.1: Histogram of the number of points in each subspace identified through the Reduced Row Echelon Form (RREF) of the TF-IDF matrix.

Next, we need an appropriate affinity measure that would allow us to merge the previously identified N_c connected components into K meaningful clusters. In this work, we utilise the concept of *principal angles*, which is first introduced in Jordan (1875).

Definition 5.4.1 (Principal Angles)**.** Let $\mathcal{S}_i$ and $\mathcal{S}_j$ be two linear subspaces of an inner product space with $1 \leqslant \dim(\mathcal{S}_i) = q_i \leqslant \dim(\mathcal{S}_j) = q_j$. The *principal angles*,

$$0 \leqslant \theta_1 \leqslant \theta_2 \leqslant \ldots \leqslant \theta_{q_i} \leqslant \frac{\pi}{2},$$

between $\mathcal{S}_i$ and $\mathcal{S}_j$ can be defined recursively for $l = 1, \ldots, q_i$ as,

$$\cos(\theta_l) = \max_{\boldsymbol{u} \in \mathcal{S}_i} \max_{\boldsymbol{v} \in \mathcal{S}_j} \cos(\boldsymbol{u}^\mathsf{T} \boldsymbol{v}), \tag{5.4.2}$$

subject to

$$\|\boldsymbol{u}\|_2 = \|\boldsymbol{v}\|_2 = 1,$$

and

$$\boldsymbol{u}^\mathsf{T} \boldsymbol{u}_m = \boldsymbol{v}^\mathsf{T} \boldsymbol{v}_m = 0, \ \forall \ 0 < m < l,$$

in which $\boldsymbol{u}_m$ and $\boldsymbol{v}_m$ are the corrseponding principal vectors that yields Eq. (5.4.2).

Applying Principal Component Analysis (PCA) (Jolliffe, 2011) to the subset of points assigned to each connected component of G, one readily obtains an orthonormal basis for each subspace. Let the columns of matrices $Q_i \in \mathbb{R}^{P \times q_i}$ and $Q_j \in \mathbb{R}^{P \times q_j}$ constitute orthonormal bases for two linear subspaces $\mathcal{S}_i$ and $\mathcal{S}_j$, respectively. The principal angles between $\mathcal{S}_i$ and $\mathcal{S}_j$ can be obtained from Singular Value Decomposition (SVD) (Björck and Golub, 1973; Drmac, 2000), $Q_i^\mathsf{T} Q_j = U \Sigma V^\mathsf{T}$, as follows

$$\theta_l = \arccos(\Sigma_{l,l}), \ l \in \{1, \ldots, q_i\}. \tag{5.4.3}$$

Principal angles ignore the difference in dimensionality between the two subspaces, which for our purposes is very important. To accommodate for this, we assume that $\mathcal{S}_i$ and $\mathcal{S}_j$ have maximum dissimilarity along the dimensions $(q_j - q_i)$. As such, we define the dissimilarity between two linear subspaces, $\mathcal{S}_i$ and $\mathcal{S}_j$ as,

$$D_{i,j} = \frac{1}{q_j} \left(q_j - q_i + \sum_{l=1}^{q_i} (1 - \cos(\theta_l)) \right),$$

$$= 1 - \frac{1}{q_j} \sum_{l=1}^{q_i} \cos(\theta_l). \tag{5.4.4}$$

To obtain the final set of K clusters, we apply the spectral clustering algorithm proposed in Ng et al. (2002). To obtain the similarity matrix W, one can use the local scaling rule proposed in Zelnik-Manor and Perona (2005) to calculate the pairwise similarity matrix W as follows

$$W_{i,j} = \exp\left\{-\frac{D_{i,j}^2}{s_i s_j}\right\}, \tag{5.4.5}$$

where s_i (s_j) is the dissimilarity between the i-th (j-th) point and its k-th nearest neighbour (the default k is set to be 8). Alternatively, one could also calculate the entries in the similarity matrix W as the average cosine of the principal angles as

$$W_{i,j} = \frac{1}{q_i} \sum_{l=1}^{q_i} \cos(\theta_l). \tag{5.4.6}$$

All the points allocated to a given subspace are assigned to the same cluster label as the subspace. Algorithm 11 outlines the steps of our proposed approach.

5.5 Experimental Results

In this section, we compare the performance of our proposed method[2] against state-of-the-art subspace clustering and standard clustering algorithms on the Amazon product names data set obtained from Harvard Dataverse website (Cavallo, 2017). This data set contains five broad product categories: Electronics, Home and appliances, Mix, Office products, and Pharmacy and health. The product names are represented using the TF-IDF representation, which contains 2921 rows (product names) and 2106 columns (unique terms).

We compare the performance of MAC with the following clustering algorithms: K-Subspace Clustering (KSC) (Agarwal and Mustafa, 2004), Low Rank Representation (LRR) (Liu et al., 2010, 2012), Sparse Subspace Clustering (SSC) (Elhamifar and Vidal,

[2]The code is available at: `https://github.com/hankuipeng/MAC`.

Algorithm 11: Minimum Angle Clustering (MAC)

```
Input  : TF-IDF matrix: $X \in \mathbb{R}^{N \times P}$
         Number of clusters: $K$
Output : Cluster labels: $\Omega = \{\omega_1, \dots, \omega_N\}$
- Compute the Reduced Row Echelon Form (RREF):
$F = \mathrm{rref}(X^\mathsf{T}) = [\boldsymbol{f}_1, \dots, \boldsymbol{f}_N]$
- Define matrix $Y$ through $Y_{i,j} = \mathbb{1}_{\{\boldsymbol{f}_i^\mathsf{T} \boldsymbol{f}_j \neq 0\}}, \ \forall\, i, j$
- Construct graph: $G$ from adjacency matrix $A = Y^\mathsf{T} Y$
- Compute connected components of $G$: $\{\mathcal{C}_1, \dots, \mathcal{C}_{N_c}\}$
for $i = 1$ to $N_c$ do
    Apply PCA to the data points in $\mathcal{C}_i$ to obtain an orthonormal basis for the $i$-th
      subspace $Q_i \in \mathbb{R}^{P \times q_i}$
    for $j = 1$ to $(i - 1)$ do
        Estimate proximity with previous subspaces through Eq. (5.4.5) or
          Eq. (5.4.6)
    end
end
- Apply spectral clustering on $W$ to obtain cluster labels for the subspaces
- Assign the same label to all the points in the same connected component as that
  of the associated subspace
```

2013), Principal Component Divisive Partitioning (PDDP) (Boley, 1998), Latent Dirichlet Allocation (LDA) (Blei et al., 2003), and Spectral Clustering (SC) (Ng et al., 2002).

KSC, LRR, and SSC are all subspace clustering algorithms, and PDDP is included as it has been developed for the purpose of partitioning documents that are embedded in high-dimensional Euclidean space. LDA is a popular topic modelling technique has been widely applied in text mining applications. Spectral clustering is a generic clustering methodology that has been successfully applied to numerous high-dimensional applications, most notably image segmentation. A further reason for including spectral clustering is that SSC, LRR, and MAC all employ SC as the last step. Thus, it is worth investigating whether the performance of these methods can be attributed to spectral clustering. We denote as SC(X) for the result obtained by appling spectral clustering to the data matrix. It is also important to check whether the information contained in the adjacency matrix a formed using RREF suffices to correctly identify the clusters. We denote as SC(A) for the result obtained by applying spectral clustering to A.

We assess the cluster performance through three external cluster evaluation measures: Purity (Zhao and Karypis, 2004), Adjusted Rand Index (ARI) (Hubert and Arabie, 1985), and Normalised Mutual Information (NMI) (Strehl and Ghosh, 2002). We also report the computational times (in seconds) for each algorithm. All experiments are run on a DELL machine with 8 CPU cores and 8 GB of RAM. The LDA algorithm is run in Python, and the remaining algorithms are run in MATLAB. Table 5.4 reports the performance of all algorithms on the Amazon data set. The best performing results are highlighted in bold, and the second best performing results are underlined.

Method	MAC	SSC	LRR	KSC
Purity	**0.78**	0.52	0.64	0.51
ARI	**0.39**	0.02	0.20	0.01
NMI	<u>0.40</u>	0.05	0.31	0.03
Runtime	157.71	413.86	534.16	40.29

Method	SC(X)	SC(A)	LDA	PDDP
Purity	0.51	0.73	0.51	<u>0.76</u>
ARI	0.04	0.24	0.02	**0.39**
NMI	0.03	**0.42**	0.05	<u>0.40</u>
Runtime	157.97	82.18	**2.15**	<u>14.33</u>

Table 5.4: Clustering performance and runtime comparison (in seconds) on the Amazon data set using TF-IDF representation.

As is shown in the table, MAC achieves the best performance in terms of Purity and ARI and second best performance in terms of NMI. With that said, the performance of PDDP is also on par with that of MAC. It is important to note that the performance of MAC is substantially better than that of SC(X), which uses the original TF-IDF matrix. Meanwhile we can see only a small performance improvement in MAC when compared to SC(A), which uses as similarity matrix the adjacency matrix A obtained after transforming the TF-IDF matrix into the reduced row echelon form. This means that the advantageous performance of MAC mainly comes from the RREF step. The third best performing algorithm is LRR with an NMI score that is above 0.3. This is not surprising, as it also utilises the low rank structure of the data. It is worth noting that its computational time is substantially higher than MAC and most other algorithms.

5.6 Conclusions & Future Work

We propose a simple algorithm for subspace clustering that is effective in clustering collections of very short texts. The algorithm is designed to exploit the properties of the very sparse and high-dimensional TF-IDF representation of such data sets. It first identifies a large number of low-dimensional linear subspaces that contain small clusters of texts which share common words. To merge them into meaningful clusters, we use principal angles to quantify the proximity between linear subspaces. Experimental results on the US Amazon data set show that this simple approach compares favourably with standard and subspace clustering methods.

In future work, we aim to identify the hierarchical structure of product categories. We also plan to investigate active learning approaches to assist the cluster validation process. Active learning aims to learn the true relationship between data objects and their labels using as least queries as possible, and involving as few data objects as possible (Settles, 2008). While there have been extensive research in active learning for classification problems (Tong and Koller, 2001; Nigam et al., 1998), active learning for clustering is an area that has been much less touched upon.

Chapter 6

Conclusions & Future Work

In this thesis, we have studied different aspects of and made contributions to subspace clustering, constrained clustering, and active learning. In closing, we present a summary of our work, based upon which we discuss a few potential avenues for future work.

Subspace Clustering with Active Learning (SCAL). In Chapter 3, we proposed a subspace clustering with active learning framework, within which we presented both an active strategy for querying informative points and a constrained clustering algorithm for incorporating the imposed constraints. The proposed framework is designed in the context of K Subspace Clustering, but the experiments on real data have shown promising results of the proposed framework when applied to spectral-based subspace methods as well.

In the proposed framework, there are two factors that are accountable for evaluating the informativeness of a point. Firstly, we evaluate the decrease in the reconstruction error if a point is removed from its assigned cluster. Secondly, we choose one out of the remaining clusters that the point has the smallest reconstruction error to, and evaluate the increase in the reconstruction error if the point is being added to the cluster. Here, picking the cluster that the chosen point has the smallest reconstruction error to is a heuristic. One principled approach could be to evaluate the influence of adding the chosen point to each of the remaining clusters, and pick the one that it has the smallest influence to.

Additionally, it would also be interesting to explore the scenario where the queried information are given in terms of pairwise ‘must-link’ and ’cannot-link’ constraints. One way to achieve this could be through the use of a certain-sample set (Xiong et al., 2017). A certain-sample set contains points that are known to belong to the same class. In Xiong et al. (2017), the proposed algorithm first picks an informative point and then queries a sequence of pairwise relationships between the informative point to a point in each of the existing certain-sample sets. Once the certain-sample sets are formed, we could still apply our proposed constrained clustering algorithm to update the current subspaces and satisfy the constraints.

Another interesting direction of research is to explore the scenario where multiple external labellers are available, and the labels or pairwise relationships that they provide do not necessarily agree with each other. Donmez et al. (2009) studied the problem of jointly learning the reliability of different labellers and choosing the most informative labels to improve the model performance in the setting of supervised learning. In the unsupervised setting, we could utilise the same idea and potentially construct a confidence score for the labellers and obtain the final label as a weighted majority vote across all labellers.

Weighted Sparse Simplex Representation (WSSR). In Chapter 4, we built a unified framework that combines subspace clustering, constrained clustering, and active learning together for spectral-based subspace clustering methods. To begin with, we proposed a spectral-based subspace clustering methodology – Weighted Sparse Simplex Representation (WSSR). It solves a quadratic programming problem that approximates each point as a convex combination of a few other points. We have also shown that a sub-problem of WSSR can be solved analytically and efficiently, which can achieve similar performance as WSSR when the noise level in the data is low. Experimental results show that WSSR is competitive against state-of-the-art spectral-based subspace clustering methods. In the unified framework, WSSR interacts with and improves upon the queried labelling information. In the first stage, the labels of the most informative point(s) are queried according to our proposed query strategy in Chapter 3. In the second stage, the constraint information are incorporated into the WSSR problem formulation through a flexible weighting scheme.

In WSSR, we used the inverse absolute cosine similarity measure for computing the entries in the weight matrix. This is a reasonable choice of proximity measure for data lying in a union of linear subspaces, and has been previously used in Heckel and Bölcskei (2015). However, as mentioned earlier, different measures need to be considered in the setting of affine subspaces.

Another interesting direction would be to explore and understand the suitability and limitations of our proposed framework in the manifold learning and clustering setting. A similar idea of approximating each point as a linear combination of its neighbouring points have been proposed in Elhamifar and Vidal (2011) for the manifold clustering setting. This is motivated by the fact that every local region on a manifold can be suitably approximated with a subspace. Therefore, there is reason to believe that our framework can lend itself naturally into the manifold learning and clustering setting with an appropriate choice of the neighbourhood size parameter.

In recent years, the development of Graphical Processing Units (GPUs) and the availability of an ever-growing amount of data have largely facilitated the rise of deep learning techniques. A number of neural network architectures have been developed and successfully applied in various application domains, most noticeably in computer vision and natural language processing (Goodfellow et al., 2016). Since what we propose is a general framework that does not depend on the data embeddings per se, it would be interesting to combine the process of training deep features with neural networks and applying our proposed method to the learnt features. In particular, it is worth exploring the possibility of training embeddings that enjoy subspace structure thus facilitating the subsequent clustering process.

Minimum Angle Clustering (MAC). In Chapter 5, we studied the problem of clustering short texts through the application of the US Amazon web-scraped text data. The data represented using the TF-IDF representation is both sparse and high-dimensional. Through exploratory data analysis, we have discovered that the variability of each sub-matrix corresponding to each product category can be summarised well in a much lower dimensional subspace. We studied and experimented with different document clustering and topic modelling techniques on the data. Additionally, we also proposed a simple subspace clustering algorithm that exploits the subspace structure that we observe in the data. The algorithm first utilises the Reduced Row Echelon Form (RREF) technique from linear algebra to first identify a large number of subspaces. To merge these subspaces into a pre-specified number of clusters, we further propose a subspace proximity measure based on the notion of principal angles.

To continue along this line of research, it would be interesting to explore other low rank matrix factorisation techniques that could potentially replace and improve the role of RREF in the first stage. For example, Liu et al. (2012) also exploits the low rankness of the data matrix through a nuclear norm minimisation programme. However, the authors directly use the coefficients of the low rank representation to form a data affinity matrix without an initial grouping. One underlying assumption of our proposed methodology is that the text data lie in a union of linear subspaces, thus the pairwise proximity can be suitably measured through principal angles. However, it may no longer be a suitable subspace proximity measure in the setting of affine subspaces. Therefore, it would be desirable to design a measure that takes into account both the angles and the displacement between subspaces (Shirazi et al., 2015). We also assumed maximum dissimilarity along the difference in the dimensions between two subspaces, which is a simple heuristic. The work of Ye and Lim (2014) provides an extensive and theoretical treatment on the dissimilarity between linear and affine subspaces with either equal or different dimensions.

To summarise, this thesis has explored in the realm of subspace clustering, and the interplay of subspace clustering with constrained clustering and active learning. Our work finds its applications in document clustering, image recognition, and motion segmentation among others. We believe that developing methodologies that are able to determine which data points are worth being queried thus validated in the model is of equal importance as designing effective and efficient algorithms in the context of clustering.

Bibliography

Agarwal, P. K. and Mustafa, N. H. (2004). K-means projective clustering. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 155–165. ACM.

Aloise, D., Deshpande, A., Hansen, P., and Popat, P. (2009). NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248.

Amelio, A. and Pizzuti, C. (2015). Is normalized mutual information a fair measure for comparing community detection methods? In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 1584–1585.

Andreev, K. and Racke, H. (2006). Balanced graph partitioning. *Theory of Computing Systems*, 39(6):929–939.

Archambeau, C., Delannay, N., and Verleysen, M. (2008). Mixtures of robust probabilistic principal component analyzers. *Neurocomputing*, 71(7-9):1274–1282.

Arias-Castro, E., Lerman, G., and Zhang, T. (2017). Spectral clustering based on local PCA. *Journal of Machine Learning Research*, 18(1):253–309.

Arthur, D. and Vassilvitskii, S. (2006). k-means++: The advantages of careful seeding. Technical report, Stanford.

Balcan, M.-F., Broder, A., and Zhang, T. (2007). Margin based active learning. In *International Conference on Computational Learning Theory*, pages 35–50. Springer.

Bartels, R. H. and Stewart, G. W. (1972). Solution of the matrix equation $AX + XB = C$. *Communications of the ACM*, 15(9):820–826.

Bartholomew, D. J., Knott, M., and Moustaki, I. (2011). *Latent variable models and factor analysis: A unified approach*, volume 904. John Wiley & Sons.

Basri, R. and Jacobs, D. W. (2003). Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 218–233.

Basu, S., Davidson, I., and Wagstaff, K. (2008). *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press.

Bellman, R. (1966). Dynamic programming. *Science*, 153(3731):34–37.

Bénasséni, J. (2018). A correction of approximations used in sensitivity study of principal component analysis. *Computational Statistics*, pages 1–17.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Björck, A. and Golub, G. H. (1973). Numerical methods for computing angles between linear subspaces. *Mathematics of Computation*, 27(123):579–594.

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022.

Boley, D. (1998). Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344.

Boult, T. E. and Brown, L. G. (1991). Factorization-based segmentation of motions. In *Proceedings of the IEEE Workshop on Visual Motion*, pages 179–180. IEEE Computer Society.

Boyd, S., Parikh, N., and Chu, E. (2011). *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Bradley, P. S. and Mangasarian, O. L. (2000). k-plane clustering. *Journal of Global Optimization*, 16(1):23–32.

Cavallo, A. (2017). Are online and offline prices similar? evidence from large multi-channel retailers. *American Economic Review*, 107(1):283–303.

Chapelle, O., Scholkopf, B., and Zien, A. (2006). *Semi-supervised learning*. MIT Press.

Chen, Y. and Ye, X. (2011). Projection onto a simplex. *arXiv:1101.6081*.

Christopher, M. B. (2006). *Pattern recognition and machine learning*. Springer.

Chung, F. R. K. (1997). *Spectral graph theory*, volume 92. American Mathematical Society.

Costeira, J. and Kanade, T. (1995). A multi-body factorization method for motion analysis. In *Proceedings of IEEE International Conference on Computer Vision*, pages 1071–1076. IEEE.

Costeira, J. P. and Kanade, T. (1998). A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179.

Cover, T. M. and Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.

Critchley, F. (1985). Influence in principal components analysis. *Biometrika*, 72(3):627–636.

Cvetković, D. M., Doob, M., and Sachs, H. (1980). *Spectra of graphs: Theory and application*. VEB Deutscher Verlag der Wissenschaften.

Davenport, M. A. and Wakin, M. B. (2010). Analysis of orthogonal matching pursuit using the restricted isometry property. *IEEE Transactions on Information Theory*, 56(9):4395–4401.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*.

Dhillon, I. S., Guan, Y., and Kulis, B. (2004). Kernel k-means, spectral clustering and normalized cuts. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 551–556. ACM.

Dhillon, I. S. and Modha, D. S. (2001). Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1-2):143–175.

Donmez, P., Carbonell, J. G., and Bennett, P. N. (2007). Dual strategy active learning. In *European Conference on Machine Learning*, pages 116–127. Springer.

Donmez, P., Carbonell, J. G., and Schneider, J. (2009). Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 259–268.

Driver, H. and Kroeber, A. (1932). Quantitative expression of cultural relationships. *University of California Publications in American Archaeology and Ethnology*.

Drmac, Z. (2000). On principal angles between subspaces of Euclidean space. *SIAM Journal on Matrix Analysis and Applications*, 22(1):173–194.

Dua, D. and Graff, C. (2017). UCI Machine Learning Repository.

Elhamifar, E. and Vidal, R. (2009). Sparse subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797. IEEE.

Elhamifar, E. and Vidal, R. (2011). Sparse manifold clustering and embedding. In *Advances in Neural Information Processing Systems*, pages 55–63.

Elhamifar, E. and Vidal, R. (2013). Sparse subspace clustering: Algorithm, theory, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2765–2781.

Enguix-González, A., Muñoz-Pichardo, J., Moreno-Rebollo, J., and Pino-Mejías, R. (2005). Influence analysis in principal component analysis through power-series expansions. *Communications in Statistics—Theory and Methods*, 34(9-10):2025–2046.

Filippone, M., Camastra, F., Masulli, F., and Rovetta, S. (2008). A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176–190.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Forgy, E. (1965). Cluster analysis of multivariate data: Efficiency versus interpretability models. *Biometrics*, 61(3):768–769.

Freund, Y., Seung, H. S., Shamir, E., and Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168.

Friedland, S. and Lim, L.-H. (2018). Nuclear norm of higher-order tensors. *Mathematics of Computation*, 87(311):1255–1281.

Gagniuc, P. A. (2017). *Markov chains: From theory to implementation and experimentation*. John Wiley & Sons.

Gaines, B. R., Kim, J., and Zhou, H. (2018). Algorithms for fitting the constrained lasso. *Journal of Computational and Graphical Statistics*, 27(4):861–871.

Gear, C. W. (1994). Feature grouping in moving objects. In *Proceedings of the IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pages 214–219. IEEE.

Gear, C. W. (1998). Multibody grouping from motion images. *International Journal of Computer Vision*, 29(2):133–150.

Girolami, M. (2002). Mercer kernel-based clustering in feature space. *IEEE Transactions on Neural Networks*, 13(3):780–784.

Goh, A. and Vidal, R. (2007). Segmenting motions of different types by unsupervised manifold clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6. IEEE.

Golub, G. H. and Van Loan, C. F. (2013). *Matrix computations*. The Johns Hopkins University Press, 4th edition.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT Press.

Gruber, A. and Weiss, Y. (2004). Multibody factorization with uncertainty and missing data using the EM algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–I. IEEE.

Guo, L. (2013). *Bayesian biclustering on discrete data: Variable selection methods*. Harvard University.

Hagen, L. and Kahng, A. B. (1992). New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085.

He, J., Zhang, Y., Wang, J., Zeng, N., and Hao, H. (2016). Robust K-subspaces recovery with combinatorial initialization. In *IEEE International Conference on Big Data (Big Data)*, pages 3573–3582. IEEE.

Heckel, R. and Bölcskei, H. (2015). Robust subspace clustering via thresholding. *IEEE Transactions on Information Theory*, 61(11):6320–6342.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Hong, W., Wright, J., Huang, K., and Ma, Y. (2006). Multiscale hybrid linear models for lossy image representation. *IEEE Transactions on Image Processing*, 15(12):3655–3671.

Hu, H., Lin, Z., Feng, J., and Zhou, J. (2014). Smooth representation clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3834–3841.

Huang, H., Yan, J., Nie, F., Huang, J., Cai, W., Saykin, A. J., and Shen, L. (2013). A new sparse simplex model for brain anatomical and genetic network analysis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 625–632. Springer.

Huang, J., Nie, F., and Huang, H. (2015). A new simplex sparse learning model to measure data similarity for clustering. In *24th International Joint Conference on Artificial Intelligence*.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1):193–218.

Hull, J. J. (1994). A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554.

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

Jolliffe, I. (2011). Principal component analysis. In *International Encyclopedia of Statistical Science*, pages 1094–1096. Springer.

Jonker, R. and Volgenant, A. (1987). A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340.

Jordan, C. (1875). Essai sur la géométrie à n dimensions. *Bulletin de la Société mathématique de France*, 3:103–174.

Kamvar, S. D., Klein, D., and Manning, C. D. (2003). Spectral learning. In *International Joint Conference of Artificial Intelligence*. Stanford InfoLab.

Kanatani, K. (1998). Geometric information criterion for model selection. *International Journal of Computer Vision*, 26(3):171–189.

Kanatani, K. (2001). Motion segmentation by subspace separation and model selection. In *Proceedings of the IEEE International Conference on computer Vision*, volume 2, pages 586–591. IEEE.

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

Larson, R. R. (2010). Introduction to information retrieval. *Journal of the American Society for Information Science and Technology*, 61(4):852–853.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, K.-C., Ho, J., and Kriegman, D. J. (2005). Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 684–698.

Li, C. and Vidal, R. (2015). Structured sparse subspace clustering: A unified optimization framework. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 277–286.

Li, C., You, C., and Vidal, R. (2017). Structured sparse subspace clustering: A joint affinity learning and subspace clustering framework. *IEEE Transactions on Image Processing*, 26(6):2988–3001.

Li, C., You, C., and Vidal, R. (2018a). On geometric analysis of affine sparse subspace clustering. *IEEE Journal of Selected Topics in Signal Processing*, 12(6):1520–1533.

Li, C., Zhang, J., and Guo, J. (2018b). Constrained sparse subspace clustering with side-information. In *2018 24th International Conference on Pattern Recognition*, pages 2093–2099. IEEE.

Li, Z., Liu, J., and Tang, X. (2009). Constrained clustering via spectral regularization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 421–428. IEEE.

Lipor, J. (2017). *Sensing structured signals with active and ensemble methods*. PhD thesis, University of Michigan–Ann Arbor.

Lipor, J. and Balzano, L. (2015). Margin-based active subspace clustering. In *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, pages 377–380. IEEE.

Lipor, J. and Balzano, L. (2017). Leveraging union of subspace structure to improve constrained clustering. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2130–2139. JMLR.

Lipor, J., Hong, D., Zhang, D., and Balzano, L. (2017). Subspace clustering using ensembles of K-subspaces. *arXiv:1709.04744*.

Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y., and Ma, Y. (2012). Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184.

Liu, G., Lin, Z., and Yu, Y. (2010). Robust subspace segmentation by low-rank representation. In *Proceedings of the 27th International Conference on Machine Learning*, pages 663–670.

Liu, H. and Fu, Y. (2015). Clustering with partition level side information. In *2015 IEEE International Conference on Data Mining*, pages 877–882. IEEE.

Liu, H., Tao, Z., and Fu, Y. (2018). Partition level constrained clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(10):2469–2483.

Lu, C., Min, H., Zhao, Z., Zhu, L., Huang, D., and Yan, S. (2012). Robust and efficient subspace segmentation via least squares regression. In *European Conference on Computer Vision*, pages 347–360. Springer.

Lütkepohl, H. (1996). *Handbook of matrices*, volume 1. Wiley Chichester.

Ma, Y., Derksen, H., Hong, W., and Wright, J. (2007). Segmentation of multivariate mixed data via lossy data coding and compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(9):1546–1562.

Ma, Y., Yang, A. Y., Derksen, H., and Fossum, R. (2008). Estimation of subspace arrangements with applications in modeling and segmenting mixed data. *SIAM Review*, 50(3):413–458.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297.

McWilliams, B. and Montana, G. (2014). Subspace clustering of high-dimensional data: A predictive approach. *Data Mining and Knowledge Discovery*, 28(3):736–772.

Melville, P. and Mooney, R. J. (2004). Diverse ensembles for active learning. In *Proceedings of the 21st International Conference on Machine Learning*, page 74. ACM.

Nasihatkon, B. and Hartley, R. (2011). Graph connectivity in sparse subspace clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2137–2144. IEEE.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856.

Nguyen, H. T. and Smeulders, A. (2004). Active learning using pre-clustering. In *Proceedings of the 21st International Conference on Machine Learning*, page 79. ACM.

Nguyen, X. V., Epps, J., and Bailey, J. (2009). Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th International Conference on Machine Learning*.

Nigam, K., McCallum, A., Thrun, S., and Mitchell, T. (1998). Learning to classify text from labeled and unlabeled documents. *AAAI*, 792.

Ostrovsky, R., Rabani, Y., Schulman, L. J., and Swamy, C. (2013). The effectiveness of Lloyd-type methods for the k-means problem. *Journal of the ACM*, 59(6):1–22.

Parikh, N. and Boyd, S. (2014). Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239.

Pati, Y. C., Rezaiifar, R., and Krishnaprasad, P. S. (1993). Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, pages 40–44. IEEE.

Peng, H. and Pavlidis, N. G. (2019). Subspace clustering with active learning. In *IEEE International Conference on Big Data (Big Data)*, pages 135–144. IEEE.

Peng, H., Pavlidis, N. G., Eckley, I. A., and Tsalamanis, I. (2018). Subspace clustering of very sparse high-dimensional data. In *IEEE International Conference on Big Data (Big Data)*, pages 3780–3783. IEEE.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

Rao, S., Tron, R., Vidal, R., and Ma, Y. (2010). Motion segmentation in the presence of outlying, incomplete, or corrupted trajectories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1832–1845.

Roy, N. and McCallum, A. (2001). Toward optimal active learning through Monte Carlo estimation of error reduction. *Proceedings of the 18th International Conference on Machine Learning*, pages 441–448.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523.

Saul, L. K. and Roweis, S. T. (2003). Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155.

Segal, M. R., Dahlquist, K. D., and Conklin, B. R. (2003). Regression approaches for microarray data analysis. *Journal of Computational Biology*, 10(6):961–980.

Settles, B. (2008). *Curious machines: Active learning with structured instances*. PhD thesis, University of Wisconsin–Madison.

Settles, B. (2009). Active learning literature survey. *University of Wisconsin-Madison*.

Settles, B., Craven, M., and Ray, S. (2008). Multiple-instance active learning. In *Advances in Neural Information Processing Systems*, pages 1289–1296.

Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by committee. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pages 287–294. ACM.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

Shi, L. (1997). Local influence in principal components analysis. *Biometrika*, 84(1):175–186.

Shirazi, S., Sanderson, C., McCool, C., and Harandi, M. T. (2015). Bags of affine subspaces for robust object tracking. In *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–7. IEEE.

Soltanolkotabi, M. and Candes, E. J. (2012). A geometric analysis of subspace clustering with outliers. *The Annals of Statistics*, 40(4):2195–2238.

Souvenir, R. and Pless, R. (2005). Manifold clustering. In *Proceedings of the 10th IEEE International Conference on Computer Vision*, volume 1, pages 648–653. IEEE.

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

Su, H., Deng, J., and Feifei, L. (2012). Crowdsourcing annotations for visual object detection. In *Workshops at the 26th AAAI Conference on Artificial Intelligence*.

Tasoulis, S. and Tasoulis, D. (2008). Improving principal direction divisive clustering. In *14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Workshop on Data Mining using Matrices and Tensors*. Citeseer.

Tasoulis, S. K., Tasoulis, D. K., and Plagianakos, V. P. (2010). Enhancing principal direction divisive clustering. *Pattern Recognition*, 43(10):3391–3411.

Tibshirani, R. J. (2011). *The solution path of the generalized lasso*. Stanford University.

Tipping, M. E. and Bishop, C. M. (1999a). Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482.

Tipping, M. E. and Bishop, C. M. (1999b). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622.

Tong, S. and Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2(Nov):45–66.

Tron, R. and Vidal, R. (2007). A benchmark for the comparison of 3-D motion segmentation algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE.

Tropp, J. A. (2004). Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242.

Tsakiris, M. C. and Vidal, R. (2017). Algebraic clustering of affine subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2):482–489.

Tseng, P. (2000). Nearest q-flat to m points. *Journal of Optimization Theory and Applications*, 105(1):249–252.

Vidal, R. (2011). Subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68.

Vidal, R., Ma, Y., and Sastry, S. (2003). Generalized principal component analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–I. IEEE.

Vidal, R., Ma, Y., and Sastry, S. (2005). Generalized principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1–15.

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

Wagner, D. and Wagner, F. (1993). Between min cut and graph bisection. *Mathematical Foundations of Computer Science*, pages 744–750.

Wagstaff, K., Cardie, C., Rogers, S., and Schrödl, S. (2001). Constrained k-means clustering with background knowledge. In *Proceedings of the 18th International Conference on Machine Learning*, volume 1, pages 577–584.

Wagstaff, K. L. (2006). Value, cost, and sharing: Open issues in constrained clustering. In *International Workshop on Knowledge Discovery in Inductive Databases*, pages 1–10. Springer.

Wang, B. and Kuo, C.-C. J. (2020). SBERT-WK: A sentence embedding method by dissecting BERT-based word models. *arXiv:2002.06652*.

Wang, D., Ding, C., and Li, T. (2009). K-subspace clustering. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 506–521. Springer.

Wang, S. and Liski, E. P. (1993). Effects of observations on the eigensystem of a sample covariance matrix. *Journal of Statistical Planning and Inference*, 36(2-3):215–226.

Wang, W. and Carreira-Perpinán, M. A. (2013). Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *arXiv:1309.1541*.

Wang, X. and Davidson, I. (2010a). Active spectral clustering. In *2010 IEEE International Conference on Data Mining*, pages 561–568. IEEE.

Wang, X. and Davidson, I. (2010b). Flexible constrained spectral clustering. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 563–572. ACM.

Wang, X., Qian, B., and Davidson, I. (2014). On constrained spectral clustering and its applications. *Data Mining and Knowledge Discovery*, 28(1):1–30.

Wang, Y., Wang, Y., and Singh, A. (2016). Graph connectivity in noisy sparse subspace clustering. In *Artificial Intelligence and Statistics*, pages 538–546.

Wilkinson, J. H. (1965). *The algebraic eigenvalue problem*, volume 87. Clarendon Press, Oxford.

Xiong, C., Johnson, D. M., and Corso, J. J. (2017). Active clustering with model-based uncertainty reduction. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 39(1):5–17.

Yang, A. Y., Rao, S. R., and Ma, Y. (2006). Robust statistical estimation and segmentation of multiple subspaces. In *Conference on Computer Vision and Pattern Recognition Workshop*. IEEE.

Yang, J., Liang, J., Wang, K., Rosin, P., and Yang, M.-H. (2019a). Subspace clustering via good neighbors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019b). XLnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, pages 5754–5764.

Ye, K. and Lim, L.-H. (2014). Distance between subspaces of different dimensions. *arXiv preprint arXiv:1407.0900*, 4.

You, C. (2018). *Sparse methods for learning multiple subspaces from large-scale, corrupted and imbalanced data*. PhD thesis, Johns Hopkins University.

You, C., Robinson, D., and Vidal, R. (2016). Scalable sparse subspace clustering by orthogonal matching pursuit. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3918–3927.

Zelnik-Manor, L. and Perona, P. (2005). Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems*, pages 1601–1608.

Zhang, D. and Balzano, L. (2016). Global convergence of a Grassmannian gradient descent algorithm for subspace estimation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 1460–1468.

Zhang, T., Szlam, A., and Lerman, G. (2009). Median K-flats for hybrid linear modeling with many outliers. In *12th IEEE International Conference on Computer Vision Workshops*, pages 234–241. IEEE.

Zhao, Y. and Karypis, G. (2001). Criterion functions for document clustering: Experiments and analysis. Technical report, University of Minnesota.

Zhao, Y. and Karypis, G. (2004). Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55(3):311–331.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.