MODEL-BASED LINEAR MANIFOLD CLUSTERING

By

Rave Harpaz

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfilment for the requirements for the degree of Doctor of Philosophy, The City University of New York

2008

UMI Number: 3296943

UMI Microform 3296943

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

THE CITY UNIVERSITY OF NEW YORK
DEPARTMENT OF
COMPUTER SCIENCE

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

	Robert Haralick
Date	Chair of Examining Committee
	Ted Brown
Date	Executive Officer

Sergei Artemov: _____

Stathis Zachos: _____

Maneesh Singh: _____

Supervisory Committee

Abstract

MODEL-BASED LINEAR MANIFOLD CLUSTERING

by

Rave Harpaz

Advisor: Professor Robert Haralick

A new paradigm of clustering called "Linear Manifold Clustering" which is based on linear manifolds is designed, analyzed, and evaluated throughout this thesis. A Linear manifold is a translated subspace. Linear manifold clustering seeks to identify groups of points that are embedded in lower dimensional linear manifolds. The "birth" of this paradigm of clustering is a consequence of what we believe is a need for an important yet overlooked cluster model, and as a result of what we identify as an acute need to sufficiently address certain clustering requirements that current state of art methods are unable to address.

In many problem domains it assumed that linear models are sufficient enough to describe and capture the data's inherent structure. Yet very few remote attempts have been made to devise clustering methods able to identify or learn mixtures of linear manifolds. None of these attempts posed the problem in a model-based statistical setting intended to model and understand the underlying "process" responsible for generating sets of points that lie on lower dimensional linear manifolds. In this thesis

we introduce a formal stochastic *linear manifold cluster model*. Based on this model we present a series of results and techniques demonstrating the applicability of the linear manifold clustering paradigm to a wide range of applications. We show that this model is a generalization of other more common and somewhat limited cluster models. This generalization allows for less assumptions to be imposed on the data, which typically yield biased results, and more freedom for the data to "speak for itself". An emphasis is put on the paradigms of *pattern* and *correlation clustering* and on the application of DNA microarray analysis, where we show that pattern clusters or correlations manifest themselves as linear manifolds in the data space. Based on the linear manifold cluster model we present two clustering algorithms: one for clustering or learning mixtures of linear manifolds, and the other tailored to the application of correlation clustering and DNA micro array analysis. The efficacy of these techniques is demonstrated by a series of experiments on synthetic and real data sets.

Most clustering methods focus only on the grouping aspects of clustering and lack the ability to provide any scientific content. In this thesis, we also present two linear manifold based modeling techniques that deliver scientific content and with which data can be described. One based on a *probabilistic density estimation* model with which statistical inference such as predictions can be based upon. The other, a model which describes the linear dependencies in the data in the form of a set of linear equations.

Acknowledgements

Completing this journey would not have been possible without the aid, support and friendship of countless many people.

First, I would like to acknowledge the source of everything in this thesis, my academic advisor Robert Haralick: his insightful ideas, contagious inspiration, and constant support had a major influence on this work and on my being throughout the past years. I am convinced that his guidance and teachings will benefit me in the future.

Second, I'd like to express my sincere gratitude to Stathis Zachos for inspiring me to join the academic world, mentoring me through my first steps at CUNY and sharing with me his worldly knowledge through many entertaining dinners. I'd like to thank Rohit Parikh for "infecting" me with the spark to aspire and to pursue knowledge in the early days. Many thanks to Keith Harrow and Brooklyn College for opening the doors of the academic world to me and providing me with the opportunities that I desperately needed when first arriving in New York. I'd also like to thank Sergei Artemov for his good advice, Ramesh Visvanathan for his counsel and the interesting conversations we shared, and Ted Brown for his support.

Third, I'd like to acknowledge my peers and fellow researches: Wenbo Cao, Christoforos Christoforou, Ingrid Montealegre, Guangmin Shi, Jose Hanchi, Alexei

Miasnikov, Jayson Rome, Eric Pacuit and Noson Yanofsky, for the great times and fruitful discussions we shared.

Fourth, I'd to thank my dear friends for all the gifts that I have received from them and for touching my life in such meaningful and profound ways: Suzanne Tamang, Lior Paster, Delaram Kahrobaei, Sabine Schumacher, Sharon Graif, Guy Dymschiz, Kaidi Song and Katia Klopfer. Special thanks need also be extended to Karen Schlain for being the best "boss" I ever had and for her endless encouragement and to Joe Driscoll for being the most generous and kind person one can ever have the opportunity to acquaint.

Last, but far from least, I'd to express the highest form of gratitude to my family: to my parents Avi and Sharit Harpaz, to my grandparents Shalom and Matilda Shirazi, and to my sisters Hili and Vered Harpaz for their endless love and support, and for suffering my faults and foibles with their infinite patience. This thesis is dedicated to them.

Computer science is no more about computers than astronomy is about telescopes.

Edsger W. Dijkstra

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Clustering is the process of grouping or classifying a collection patterns, into classes called *clusters* so that the patterns within a cluster are "similar" to one another, yet "dissimilar" to patterns in other clusters.

Technology advances in the past few decades have made it possible to collect and organize huge amounts of transactional and experimental data. This in turn has created a need for more advanced and versatile *data analysis* techniques. At a broad level, data analysis can be categorized as either exploratory in nature or confirmatory. The key to both is the classification of measurements based on either natural groupings (clusters) revealed through analysis, or on a *goodness of fit* to a postulated model. Hence, from a data analysis perspective clustering may be considered an exploratory process.

Machine learning is the study and design of computer programs that are able to learn patterns, regularities, or rules from past observations. The two major types of learning techniques are typically referred to as *supervised* and *unsupervised*. In

supervised learning we are provided with a collection of labeled (pre-classified) patterns, which are used to learn class descriptions or classification rules. In the case of unsupervised learning we are provided with a collection of unlabeled (unclassified) patterns, and attempt to learn structure, or describe the data in some meaningful way. One way of doing so is to organize or partition the data into homogenous groups or classes called clusters. In a sense, these clusters are associated with labels, but the labels are learned by a data driven process and not given in advance. From a machine learning perspective clustering is considered the most important unsupervised learning problem.

Data mining is the process of analyzing data to discover previously unknown interesting and implicit knowledge in the form of regularities. This is accomplished through an analysis of patterns that form in the data. Data mining is an exploratory process, so clustering methods are well suited for data mining. Natural groupings among the data objects are probably the most important form of regularities, and from a data mining perspective the usually preliminarily task of clustering is considered the most important.

In many applications of pattern or data analysis there is little prior information (e.g., statistical models) available about the data, and the analyst must make as few as possible assumptions about the data. It is precisely under these circumstances that clustering is particularly useful for the exploration of relationships among the data objects and to make an assessment about their structure. Traditionally clustering has been used for: *data reduction*, where instead of processing the data set as a whole, cluster representatives are used; *hypothesis generation*, where cluster analysis is used to infer some hypotheses about the data, e.g., we may find that in a retail

database there are two significant groups of customers based on their age and the time of purchases, and form the hypothesis that "young people go shopping in the evening" and "old people go shopping in the morning"; *hypothesis testing*, where cluster analysis is used to validate a specific hypothesis such as the one posed above; *prediction*, where cluster analysis is applied to the data whereupon unknown patterns are classified into specified clusters based on their similarity to the clusters' features. For example, clustering is applied to a data set of patients infected by the same disease and their reaction specific drugs. Based on this clustering we can then assign a new patient to a cluster which he/she can be classified and prescribe his/her medication.

Due to new increasingly important application domains, clustering has become one of the most prominent tools in data analysis and data mining. In *business and marketing* clustering may help marketers discover significant groups in their customers' database and characterize them based on their past purchasing patterns. A relatively new and growing field in which clustering is applied is *recommendation* or *collaborative filtering systems*. In this case, sets of customers with similar interest patterns (e.g., interest in books, music, movies) need to be identified. In *biology* clustering is used to define taxonomies, e.g., classification of plants and animals based on their features. Due to the completion of the *human genome project*, one of the most important applications of clustering is in the vastly growing field of *gene expression microarray analysis*. In *machine vision*, clustering is used for image segmentation or the classification of image pixels. In *Web* and *text mining*, clustering is used to discover significant groups of documents on the Web which later assists and enhances the information retrieval process.

The variety of clustering applications, goals, cluster models, and techniques for

representing data, measuring similarity, and grouping data has lead to a very rich assortment of clustering methods and clustering schemes. Earlier or "classical" clustering methods typically dealt with smaller sets of features (lower dimensional data) and considered all the features of the data simultaneously relevant to each of the underlying clusters of the data. That is, each cluster identified by a clustering algorithm contained a subset of points each represented by the full set of features or dimensions underlying the data. Because of this, the name "full-dimensional" or "full-space" clustering has often been associated with these clustering methods. However, due to recent technology advances in data collection many applications of clustering are now characterized by high dimensional data. These type of data sets pose new challenges which the full-dimensional clustering methods are unable to address. First, because of the relatively large feature set, very often not all features of the data are relevant to the clustering, and the presence of non-information carrying or irrelevant noisy features has the potential to eliminate clustering tendency and mislead the clustering algorithm by masking clusters in noisy or irrelevant data. In other words, clusters are hidden and may be visible only in subsets of features or subspaces of the data. Second, is the so called "curse of dimensionality", which suggests that the sparsity of the data increases exponentially with the dimensionality of the input space given a constant amount of data, with points tending to become equidistant from one another at a certain high dimension. This in turn implies that learning structure in high dimensional spaces by distance measures that utilize the full set of dimensions becomes increasingly difficult. A recent advancement in the area of clustering was the introduction of *subspace clustering* methods designed to address these challenges. Subspace clustering seeks to identify different clusters embedded in different subspaces of the

same data set. A subspace cluster consists of a subset of points and a corresponding subset of attributes, such that these points form a dense region in a subspace defined by the set of corresponding attributes. In recent years, due to immense research efforts directed at *DNA microarray analysis* and due to the growing popularity of text mining and recommendation systems, a new paradigm of clustering has evolved. This paradigm of clustering often associated with the names *pattern* or *correlation* clustering has shifted the focus from the identification of objects with similar values to the detection of objects with similar *behavior patterns* or objects that induce large correlations among some subset of features. Traditional clustering methods including those used in subspace clustering focus on grouping objects with similar values. They define object similarity by the "physical" distance between the objects over all or a subset of dimensions, which in turn may not be adequate to capture correlations in the data or coherent behavior patterns. A set of points may be located far away from each other yet manifest coherent behavior patterns or induce large correlations. The detection of correlations is a an important data mining task because correlations may reveal a dependency or some cause and effect relationship between the features under consideration. Another important application of correlations is in data modeling where correlations may be used to carry out (local) dimensionality reduction by eliminating correlated (redundant) features. In many studies these correlations were often discussed and presented in terms of the behavior patterns objects manifest, hence the name pattern clustering often associated with methods amid at this type of problem. In gene expression microarray clustering the goal is to identify groups of genes that exhibit similar expression patterns under some subset of conditions (dimensions), from which gene function or regulatory mechanisms may be inferred.

In recommendation or collaborative filtering systems, sets of customers with similar interest patterns need to be identified so that customers' future interests can be predicted and proper recommendations be made. However, from a correlation point of view objects exhibiting coherent behavior patterns induce large correlations among their defining features. Hence, the identification of large correlations is a means by which the so called pattern clusters can also be discovered.

Despite the versatility and wide range of clustering techniques, it is well accepted that there is no clustering technique that is universally applicable in uncovering the variety of structures that may be present in various data sets or that is independent of the final aim of the clustering. In practice each clustering technique makes implicit assumptions about the shape of the clusters, the similarity criteria that is most appropriate to reveal the clusters, and the grouping technique that will be able to reveal them in the most accurate and efficient way. Needless to say, each clustering technique has its strengths and weaknesses primarily depending on the distribution of patterns in the data and the size of the data. Consequently, the underlying theme of operation has become that the user or analyst will choose the method he believes best fits the task, aims, and data at hand. The more knowledge the user has on the clustering technique, the data generation process that produced the data at hand, and the application domain, the more likely he will be able to succeed in assessing the true underlying structure of the data at hand. Because of this clustering is often considered a subjective process; the same data set often needs to be partitioned differently for different applications.

Since clustering algorithms define clusters that are not known a priori, irrespective of the clustering methods, the final partition of data requires some kind of evaluation.

After all, most clustering algorithms will, when presented with data produce clusters regardless of whether the data contains clusters or not. If the data does contain clusters, some clustering algorithms may obtain better clusters than others. *Cluster tendency* is the assessment of whether or not there is any merit to cluster analysis prior to the clustering. Data sets which do not contain clusters should not be processed by a clustering algorithm.

Cluster validity on the other hand is the evaluation of the clustering algorithm's output. Cluster validity techniques usually rely on some optimality criteria, or statistical methods, and are used to determine whether the output is meaningful. A clustering is valid if it cannot reasonably have occurred by chance or by an artifact of a clustering algorithm. There are three types of validation techniques. An *external* assessment of validity compares the recovered structure to an a priori known structure or "ground truth". An *internal* examination of validity tries to determine if the structure is intrinsically appropriate for the data. A *relative* test compares two structures and measures their relative merit.

Many validity techniques and measures have have been proposed for evaluating clustering quality when an external assessment is not possible. However, just like the clustering algorithms themselves they rely on certain assumptions or heuristic criteria which may not always be appropriate, and which may yield different results for the same partitioning. It has been suggested that validity measures should be posed in a probabilistic setting which assess the inference capabilities and scientific content delivered by the clustering. The derivation of such validity measures is beyond the scope of this thesis. Most of the cluster validation done in this thesis, evaluates or demonstrates the efficacy of the proposed methods on external validation techniques

which compares the clustering results to ground truth.

1.2 Problem Statement and Contributions

Clustering techniques should satisfy various requirements ranging from scalability, through ability to deal with a wide variety of cluster models, to interpretability of the clustering results and the delivery of scientific content. New clustering paradigms or techniques are "born" either due to new application and problem domains, or due to the need for newer cluster models, or because state of art methods are unable to sufficiently address certain clustering requirements that become more acute. In this respect the contributions made in this thesis are a consequence of what we believe is a need for an important yet overlooked cluster model, and as a result of what we identify as an acute need to sufficiently address certain clustering requirements that current state of art methods are unable to address.

Throughout this thesis we will present and discuss a new clustering paradigm, which we call "Linear Manifold Clustering", and which is based on the concept of linear manifolds. Avoiding unnecessary mathematical abstraction, a linear manifold is simply a translated subspace, which can be visualized as a line, plane, hyperplane, etc., depending on its dimensionality. In many cases and problem domains, not only in clustering, it is assumed that the data follows a linear structure, or that a linear model is sufficient enough to describe and capture the data's inherent structure. Typical examples include linear regression, PCA, and subspace clustering, which are all special cases of linear manifolds. Very few indirect attempts have been made to devise clustering methods able to identify or learn mixtures of linear manifolds, and even more so to formulate a mathematically rigorous linear manifold cluster model.

Most attempts have relied on the subspace cluster model, which as mentioned is a special case of a linear manifold.

In thesis we introduce a formal "stochastic linear manifold cluster model" which describes a "process" that generates sets of points that are embedded in or fit bounded linear manifolds. The model consists of two parts: a *deterministic* part describing the "signal", which in our case is the linear manifold which the ideal points fit, and a *stochastic* part describing the distribution of points on the manifold and their deviation from the manifold or the "noise", in the *signal processing* jargon. In the context of clustering the model proposed has the following properties: the points in each cluster are embedded in a lower dimensional linear manifold of finite extent, the intrinsic dimensionality of the cluster is the dimensionality of the linear manifold and is generally much smaller than the dimensionality of the data, the manifold is arbitrarily oriented, in the orthogonal complement space to the manifold the points form a compact densely populated region, the cluster induces correlations or linear dependencies between the attributes of the data.

Based on this model, hence the term "model-based" appearing in the title of this thesis, we present two linear manifold clustering algorithms along with several formal results and extensions demonstrating the applicability of the linear manifold clustering paradigm to a wide range of applications. Both algorithms utilize a *stochastic-model-fitting* approach. The first of the two is an algorithm designed to identify clusters embedded in lower dimensional linear manifolds, or learn probabilistic linear manifold mixture models. The second of the two algorithms is a derivative of the first that is based on 1D-linear manifolds, yet substantially different that also utilizes *feature selection* and *random walk* techniques, and is tailored to fully exploit the potential

of linear manifolds to the application of pattern/correlation clustering and DNA microarray analysis. Each of the methods discussed in this thesis is accompanied by experiments on real as well as synthetic data sets demonstrating its potential and efficacy, with an emphasis towards the end of the thesis on DNA microarray analysis. In addition to the two algorithms we also present two modeling techniques that deliver scientific content and with which data can be described. One is based on a probabilistic density estimation model, which statistical inference such as predictions can be based upon. The other is a model which describes the linear dependencies in the data in the form of a set of linear equations.

In the following we briefly outline the contributions and advantages of the linear manifold clustering paradigm.

- Classical clustering algorithms are based on the concept that a cluster center is a single point, and that clusters are sets of points compact around this central point. Most approaches to clustering neglect to consider the possibility that a cluster center may be associated with a more complex structure. The linear manifold clustering paradigm introduces the concept of linear manifold clusters which are groups of points compact around a linear manifold.

- As an exploratory process, a clustering method should make as few as possible assumptions pertaining to the shape of clusters and their underlying distribution. The linear manifold cluster model proposed in this thesis is a generalization of many other more specific cluster models, i.e., other cluster models such as the full-dimensional, subspace, and pattern or correlation cluster models are all instances of the linear manifold cluster model. This unification of clustering paradigms under one model provides a more general framework in which cluster

analysis may be performed. That is, a particular cluster structure need not be imposed on the data by a clustering algorithm any more, in a sense allowing for fewer assumptions to be made and more freedom for the data to "speak for itself". Imposing structure on the data typically leads to biased results. In the case of pattern clustering or more specifically gene expression clustering this matter was stated more drastically; it has been suggested that other types of information carrying patterns which are also instances of linear manifolds are completely overlooked by most clustering methods, and that current state of the art algorithms are not flexible enough to mine different patterns simultaneously.

- A good clustering scheme is one which provides "scientific content", that is, a probability model which describes the underlying population, and on which statistical inference such as predictions can based. One of the ultimate goals of clustering is not only to reveal structure but to understand the underlying mechanism or "process" which generated the structure. In other words, model the population of points that may have formed the structure discovered by a clustering algorithm. Describing the population by a model based on the sample on which the clustering is applied permits us gain insights and to learn the most important aspects of the population. Such a model should also provide us the capability of making predictions that are consistent with the data. Lacking scientific content clustering is merely a visualization tool providing different views of the data. Most clustering methods focus only on the grouping and lack the ability to provide any scientific content. The linear manifold clustering paradigm proposed in this thesis is essentially a model-based clustering paradigm and as such "enjoys" the advantages model-based methods have over

other clustering paradigms. Because of its solid mathematical foundation the clustering results are easer to interpret. A mixture of individual linear manifold models can easily be constructed to describe the data, the process generating it, and to provide an estimate of the underlying distribution of the population. Based on this model probabilistic classifiers can be built, and hence predictions (classification) for new points can be made. These predictions can also serve as a basis for *fuzzy* or *soft* clustering. Because of the probabilistic setting on which the linear manifold clustering paradigm is based, statistical methods such as hypothesis testing, permutations tests, maximum likelihood estimation, etc., can be applied to cluster validity problems making them more tractable.

- Common to all forms of linear correlation and linear dependencies, is that in the data space they manifest themselves as lines, planes, and generally speaking as linear manifolds. Hence, the detection of linear manifolds is a means by which correlations or linear dependencies may be identified.

- Dimensionality is a fundamental challenge in many machine learning, pattern recognition, and data mining applications. This is primarily due to the "curse of dimensionality". The aim of dimensionality reduction is to reduce or transform high dimensional data into lower dimensional data while retaining most of the underlying structure in the data, and by that circumvent the problems associated with high dimensional data. Dimensionality reduction is also used to visualize high dimensional data in a lower dimensional space and by that gain more insight to the problem at hand. Very often observed real data is a consequence of a process governed by a small number of factors. In the data space this is manifested by the data points lying or being located close to surfaces

such as linear manifolds whose intrinsic dimensionality is much smaller than the dimensionality of the data. Most dimensionality reduction techniques are "global" in the sense that they assume the whole data set can be characterized by one lower dimensional surface, and overlook the possibility that different portions of the data lie on different lower dimensional surfaces, or might be generated by a process consisting of combination of simpler processes, i.e., a mixture of models. From a dimensionality reduction point of view the linear manifold paradigm of clustering can be thought of as a dimensionality reduction technique that assumes that the data lies on different linear surfaces whose intrinsic dimensionality is much smaller than that of the data. Linear manifold learning as a dimensionality reduction technique is part of and a special case of a relatively new field of research that is growing in popularity called *Manifold learning*.

1.3 Thesis Organization

In chapter 2 we provide the notation that will be used throughout the thesis, in addition to more formal definitions of subspaces and linear manifolds. In chapter 3 we introduce and discuss the basic concepts of clustering, including: applications, components, requirements, similarity measures, cluster models, and clustering schemes. In chapters 4-6 we review different clustering schemes and paradigms, highlighting the underlying techniques, application domains, strengths, and weaknesses of each. Chapter 4 covers the full-dimensional clustering paradigms, which have laid the foundations in the area of clustering and of more sophisticated clustering techniques. The chapter will cover: *hierarchical*, *partitional*, *density*, *grid*, *fuzzy*, *model-based*, *ANN*,

and *graph* based methods. Chapter 5 discusses the concept of subspace clustering, and reviews the bottom-up and top-down subspace methods. Chapter 6 covers the concept of pattern and correlation clustering with an emphasis on gene expression microarray analysis. Chapter 7 is where we start presenting the contributions made in this thesis. In chapter 7 we present the linear manifold cluster model accompanied by an elaborate discussion and its motivating reasons. In chapter 8 we present the linear manifold clustering (learning) algorithm called LMCLUS. The chapter will cover details of its underlying principles, techniques, and statistical properties, along with a formal complexity analysis and an empirical evaluation. In chapter 9 we present extensions of linear manifold clustering to *data modeling*, focusing on *probability density estimation*, *probabilistic classification*, and *linear dependency* modeling, along with a classifier construction example and experiment. In chapter 10 we present several formal results tying linear manifold clustering to the application of pattern and correlation clustering, accompanied by experiments on gene expression data demonstrating the potential of these results. The second clustering algorithm is a specialized form of the general algorithm designed to find line (correlation) clusters in subspaces of the data. It is called SLCLUS, and is discussed in chapter 11. The algorithm is covered in the same manner in which LMCLUS is covered, yet the experimental section focuses on gene expression clustering. Finally, in the conclusion we summarize the main results and contributions made in this thesis and discuss future directions of research.

Chapter 2

Definitions and Notation

Unless stated otherwise the following terms and notation will be used throughout this thesis.

- $\mathbf{x}$ (or any lower case boldface English latter) denotes a *pattern*, *feature vector*, *data point*, *data object*, or *observation*.
- A pattern $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ is a vector of d measurements, where each measurement is called a *feature* or *attribute*. The value of the i-th feature is denoted by x_i.
- d denotes the *dimensionality* of the feature (pattern) space.
- $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ denotes a *data* or *feature set*. $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ denotes the i-th data point or feature vector in X, and x_{ij} the value of the j-th feature of $\mathbf{x}_i$.
- N denotes the *size* or number of points in the data set.
- A cluster (set of points) is denoted by C.
- $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$ denotes a set of clusters.

- K denotes the number of clusters in a data set.
- A lower case English latter such as c denotes a scaler.
- $\mathbf{x}'$ denotes the *transpose* of vector $\mathbf{x}$.
- $\mathbf{1}_k$ denotes a k dimensional vector of ones (1).
- I_k denotes a k dimensional identity matrix.
- $\|\mathbf{x}\|$ denotes the *norm* (L_2-norm) of vector $\mathbf{x}$, and $\|\mathbf{x}\|_p$ the *p-norm* of $\mathbf{x}$.
- Greek letters will be typically used to denote statistical parameters, e.g., μ denotes the mean, σ^2 variance, and Σ a covariance matrix.
- $p(x)$ denotes a *probability density function* (pdf), and $P(x)$ a (discrete) *probability function.*
- $\mathrm{E}[X]$ denotes the *expected value* of random variable (or vector) X, $\mathrm{Var}[X]$ denotes the *variance* of X, and $\mathrm{Cov}(X, Y)$ the covariance of random variables X and Y.
- $N(\mu, \sigma^2)$ denotes a *normal* (Gaussian) distribution with parameters μ and σ, $U(a, b)$ a *uniform* distribution with parameters a and b, and χ^2_k a *chi-squared* distribution with k degrees of freedom.

In the following the we restrict our attention to the finite space $\mathbb{R}^d$.

Definition 2.0.1 (Subspace)**.** *A subspace V is a nonempty set of points (vectors) that is closed under linear combination (addition, and scaler multiplication). Because of the closure property the vector $\mathbf{0}$ (zero) is included in the subspace. The dimension of V is the number of linearly independent (basis) vectors that span V.*

Definition 2.0.2 (Linear Manifold)**.** *L is a **linear manifold** of vector space V if and only if for some subspace S of V and translation $\mathbf{t} \in V$, $L = \{\mathbf{x} \in V | for\ some\ \mathbf{s} \in S, \mathbf{x} = \mathbf{t} + \mathbf{s}\}$. The dimension of L is the dimension of S, and if the dimension of L is one less than the dimension of V then L is called a **hyperplane**. A linear manifold L is rectangularly bounded if and only if for some translation $\mathbf{t}$ and bounding vectors $\mathbf{a}_L$ and $\mathbf{a}_H$, $L = \{\mathbf{x} \in V | for\ some\ \mathbf{s} \in S, \mathbf{a}_L \leq \mathbf{s} \leq a_H, \mathbf{x} = \mathbf{t} + \mathbf{s}\}$. A rectangularly bounded linear manifold has finite extent and is localized with center $\mathbf{t} + \frac{\mathbf{a}_L + \mathbf{a}_H}{2}$. In the case that $\mathbf{a}_L = -\mathbf{a}_H$, its center is the translation $\mathbf{t}$.*

From the definitions above it is clear that linear manifold is essentially a subspace that may have been translated away from the origin. A subspace is a special case of a linear manifold that contains the origin. Geometrically, a 1D manifold can be visualized as a line embedded in the space, a 2D manifold as a plane, and a 0D manifold as a point, that do not necessarily pass through the origin.

Chapter 3

Clustering Overview

3.1 What is Clustering?

Clustering is the process of grouping or classification of a collection objects or patterns, usually represented as a vector of measurements, into classes or *clusters* so that the patterns within a cluster are "similar" to one another, yet dissimilar to patterns in other clusters.

Technology advances in the past few decades have made it possible to collect and organize huge amounts of transactional and experimental data. This in turn have created the need for more advanced and versatile *data analysis* techniques, one of which is clustering. At a broad level, data analysis can be categorized as either exploratory in nature and typically used for hypothesis formation, or confirmatory in nature and typically used for decision making. Key to both is the classification of measurements based on either a "goodness of fit" to a postulated model, or natural grouping (clustering) revealed through analysis. Hence, from a data analysis perspective clustering may be considered an exploratory process.

Machine learning is the study and design of computer programs that are able to induce (learn) patterns, regularities, or rules from past observations. The two major

types of learning techniques are typically referred to as *supervised* and *unsupervised*. In the context of classification, in supervised learning we are provided with a collection of labeled (pre-classified) patterns, which are used to learn class descriptions or classification rules, which are ultimately used to classify or label an unobserved newly encountered pattern. In the case of unsupervised learning we are provided with a collection of unlabeled (unclassified) patterns, and attempt to learn structure, or describe the data in some meaningful way. One way of doing so is to organize or partition the data into homogenous groups or classes called clusters. In a sense, these clusters are associated with labels, but the labels are data driven and not given in advance. That is, the relationship between the patterns or structure of the patterns learned in the process is solely obtained from the data. In addition, clustering produces initial classes which can then be used in a supervised classification process. From a machine learning perspective clustering is considered the most important unsupervised learning problem.

Data mining or *knowledge discovery in databases* (KDD) is the process of analyzing data to discover previously unknown interesting and implicit knowledge in the form of regularities or relationships in databases [1]. This is accomplished through an analysis of patterns that form in the data. Data mining is an exploratory process, so clustering methods are well suited for data mining. Natural groupings among the data objects is probably the most important form of regularities or patterns sought in a data mining process, and from a data mining perspective the usually preliminarily task of clustering is considered the most important.

It is well accepted that there is no clustering technique that is universally applicable in uncovering the variety of structures that may be present in various data sets

or that is independent of the final aim of the clustering. In addition to that, the variety of techniques for representing data, measuring similarity, and grouping data has lead to a very rich assortment of clustering methods. In practice each clustering technique makes implicit assumptions about the shape of the clusters, the similarity criteria that is most appropriate to reveal the clusters, and the grouping technique that will be able to reveal them in the most efficient way. Needless to say, each clustering technique has its advantages and disadvantages primarily depending on the distribution of patterns in the data and the size of the data. As a consequence it is the user who must decide on the various clustering criteria and clustering technique in order that the result of the clustering will best suit their needs. Moreover, the more knowledge the user has on the clustering technique, the data generation process that produced the data at hand, and the application domain, the more likely he will be able to succeed in assessing the true underlying structure of the data at hand [2]. Because of this clustering is often considered a subjective process; the same data set often needs to be partitioned differently for different applications.

3.2 Applications

In many applications of pattern or data analysis there is little prior information (e.g., statistical models) available about the data, and the analyst must make as few as possible assumptions about the data. It is precisely under these circumstances that clustering is particularly useful for the exploration of relationships among the data objects and to make an assessment about their structure. In the following we first summarize the basic directions in which clustering is used [3], and describe some of the more prominent applications where clustering has been employed.

3.2.1 Basic Directions of Use

1. **Data reduction** - In many cases, the amount of available data is very large and its processing becomes very demanding. Clustering can be used to partition a data set into a number of interesting clusters. Then, instead of processing the data set as a whole, we adopt the representatives of the defined clusters and by that achieve data compression.

2. **Hypothesis generation** - Cluster analysis may be used in order to infer some hypotheses about the data. For instance we may find in a retail database that there are two significant groups of customers based on their age and the time of purchases. Then, we may infer some hypotheses about the data, for example, young people go shopping in the evening, old people go shopping in the morning.

3. **Hypothesis testing** - In this case, cluster analysis may be used for the verification of the validity of a specific hypothesis. For example, assuming the following hypothesis: Young people go shopping in the evening. One way to verify whether this is true is to apply cluster analysis to a representative set of stores. Suppose that each store is represented by its customers' details (age, job etc) and the time of transactions. If after applying cluster analysis a cluster that corresponds to "young people buying in the evening" is formed, then the hypothesis is supported by cluster analysis.

4. **Prediction based on groups** - Cluster analysis is applied to the data set and the resulting clusters are characterized by the features of the patterns that belong to these clusters. Then, unknown patterns can be classified into specified clusters based on their similarity to the clusters' features. Assume, for example,

that cluster analysis is applied to a data set concerning patients infected by the same disease, and that the result is a number of clusters of patients according to their reaction to specific drugs. Then for a new patient, we identify the cluster in which he/she can be classified and based on this decision his/her medication can be prescribed.

3.2.2 Specific Applications

1. **Business and Marketing** - Clustering may help marketers discover significant groups in their customers' database and characterize them based on their past purchasing patterns. Insurance companies use clustering to identify groups of motor insurance policy holders with a high average claim cost, and also to identify frauds. In city-planning the goal is identify groups of houses according to their house type, value and geographical location. A relatively new and growing field in which clustering is applied is *recommendation* or *collaborative filtering systems*. In this case, sets of customers with similar interest patterns (e.g., interest in books, music, movies) need to be identified so that customers' future interests can be predicted and proper recommendations be made.

2. **Biology** - One use of clustering in biology is to define taxonomies, e.g., classification of plants and animals based on their features. Due to the completion of the *human genome project*, one of the most important applications of clustering is in the vastly growing field of *gene expression microarray analysis*. In this case, the goal is to identify groups of genes that exhibit similar expression (behavior) patterns under some subset of conditions (tissue samples or time points) from which gene function or regulatory mechanisms may be inferred.

3. **Machine vision** - In this case clustering is primarily used for image segmentation or the classification of image pixels. However, clustering is also used as a preprocessing stage to identify pattern classes for subsequent supervised classification. In particular, clustering is used in one of the preliminary stages of object and character recognition systems.

4. **Web and text mining** - Here clustering is used to discover significant groups of documents on the Web. This classification of Web documents later assists and enhances the information retrieval process. Clustering is also used to cluster Web-log data in order to discover groups of similar access patterns. Libraries use clustering to group books, journals, and other documents as part of an information retrieval system.

3.3 Components of a Clustering Task

The typical steps in a clustering process are as follows [1, 4]:

1. **Feature representation** - The overall goal of this step is to organize the observations and create a set of feature vectors which will be input to the clustering algorithm. Underlying this general task are several smaller tasks such as:

    - The determination of feature or attribute types. Feature types are either *quantitative* (e.g., height) or *qualitative* (e.g., color). These two feature types can further be subdivided into the following subtypes: continuous (e.g., height), discrete (e.g., number of customers), and interval (e.g., duration of an event) as quantitative subtypes. And, nominal (e.g. color),

and ordinal (e.g., rank) as qualitative subtypes. After determining the feature types, their scale (for quantitative subtypes) and number needs to be determined.

- *Feature selection*, where the goal is to select a subset of the most informative and effective features from the original full set features that are relevant to the clustering.
- *Feature extraction*, where the goal is to compute a new set of features from the original set by various transformations such as *principle component analysis* (PCA) [5] that will ultimately improve clustering quality.

The user's role in the feature representation phase is critical. He must gather facts and conjecture about the data in order to create the best possible set of features for the clustering task at hand. A careful investigation of the data and suitable feature transformations or selections can significantly improve the clustering results.

2. **Clustering** - This step refers to the choice of the similarity measure, clustering criterion, and the clustering algorithm that best suits the data set at hand. Typically, the similarity measure and clustering criterion characterize the clustering algorithm.

   - The similarity or distance measure is in most cases applied on two objects and is supposed to quantify how "similar" the two objects or their corresponding features vectors are with respect to one another. Typical similarity measures should make sure that each of the individual features constituting a feature vector contribute equally to the computation of the

similarity and that there are no features that dominate others. Similarity measures are discussed more thoroughly in section 3.5.

- The clustering criterion is usually expressed as a cost function, e.g., the sum of squared distances. The clustering criterion should be chosen to account for the type or shape of clusters we expect to encounter in the data set. For most clustering algorithms the clustering criterion characterizes the type of clusters that will be produced by the clustering algorithm.
- The clustering itself can performed in many ways. As mentioned earlier the similarity and clustering criterion usually characterize the algorithm that will be used to cluster the data. Nonetheless, more decisions need to be made. Among those decisions are the determination of object membership type. In some cases it is required that the generated clusters be "hard", that is, a discrete partitioning of the data set where each object belongs to one cluster only. In other cases it is required that the generated clusters be "soft", that is, an object may belong to more than one cluster, yet another possibility is a "fuzzy" partitioning of the data, where each object has a degree of membership to each cluster. Another decision that needs to made pertains to the manner in which the results are presented. For example, whether the output should be organized in a nested hierarchy of clusters as opposed to one partitioning of the data, or whether a probabilistic description of the underlying distribution of the objects is required. Another decision that needs to made pertains to the time we are willing to allocate for the clustering task. Usually there will be a tradeoff between speed and clustering quality.

Clustering criteria and algorithms are discussed throughout chapters 4 - 6.

3. **Validation of results** - Since clustering algorithms define clusters that are not known a priori, irrespective of the clustering methods, the final partition of data requires some kind of evaluation. After all, all clustering algorithms will, when presented with data produce clusters regardless of whether the data contains clusters or not. If the data does contain clusters, some clustering algorithms may obtain better clusters than others. *Cluster tendency* is the assessment of whether or not there is any merit to cluster analysis prior to the clustering. Data sets which do not contain clusters should not be processed by a clustering algorithm. In a sense, cluster tendency is an assessment of the data domain rather than the clustering algorithm. Cluster validity on the other hand is the evaluation of the clustering algorithm's output. Cluster validity techniques usually rely on some optimality criteria, or statistical methods, and are used to determine whether the output is meaningful. A clustering is valid if it cannot reasonably have occurred by chance or is an artifact of a clustering algorithm. There are three types of validation techniques [6]. An *external* assessment of validity compares the recovered structure to an a priori structure. An *internal* examination of validity tries to determine if the structure is intrinsically appropriate for the data. A *relative* test compares two structures and measures their relative merit.

4. **Interpretation of the results** - In many cases, the experts in the application area have to integrate the clustering results with other experimental evidence and analysis in order to draw the right conclusion.

3.4 Requirements and Challenges

The main requirements that a clustering algorithm should satisfy are:

1. **Scalability** - Clustering algorithms must be efficient in order to scale well with the increasing size and dimensionality of data sets.

2. **Cluster shapes** - Because very little is known in advance about the patterns that are sought to be uncovered, clustering algorithms should be able to deal with a wide variety of cluster shapes (further discussed in section 3.6).

3. **Cluster membership** - Clustering algorithms should if required allow for the possibility that clusters may overlap, i.e., allow for a data object to be a member of several and not only one cluster.

4. **Attribute types** - Clustering algorithms should be able to deal with different types of attributes.

5. **Noise** - Clustering algorithms should be able to deal with noise and outliers effectively.

6. **High Dimensions** - Clustering algorithms should be able to deal with the problems associated with high dimensional data. Namely, the curse of dimensionality and the presence of irrelevant features (further discussed in chapter 5).

7. **Order of input** - Clustering algorithms should be insensitive to the order of input.

8. **Domain knowledge** - Clustering algorithms should require the least possible domain knowledge in order to be operated.

9. **Input parameters** - The input parameters to the algorithm should be intuitive and easy to set, and should not require extensive domain knowledge.

10. **Interpretability** - The output clusters produced by a clustering algorithm should be represented in an interpretable, usable and meaningful way.

Current state of the art clustering algorithms do not address all the requirements adequately or concurrently. In practice most methods make certain assumptions a priori, typically about the type of clusters expected to be found, and choose to focus on a subset of the requirements while trading off the others in order to address those chosen adequately. The underlying theme of operation is that the user will choose the method he believes best fits the task, aims, and data at hand.

3.5 Similarity Measures

Fundamental to the definition of a cluster and to the clustering algorithm is the similarity measure which is used to quantify the similarity between two objects or features of the same feature space, and which serves as criteria for grouping or separating items. Because of the variety of cluster types, feature types and scale types, the similarity measure also called a *distance function*, must be chosen carefully. In many applications it is also required that the distance function be a *metric*. A metric is a distance function $d(x, y)$ that satisfies the following four conditions:

1. $d(x, y) \geq 0$ (non-negativity)

2. $d(x, y) = 0 \Leftrightarrow x = y$ (identity)

3. $d(x, y) = d(y, x)$ (symmetry)

4. $d(x, z) \leq d(x, y) + d(y, z)$ (triangular inequality).

Most clustering algorithms operate on continuous feature vectors. As a consequence in the following we discuss some of the more prominent similarity measures which operate on continuous features, and use $\mathbf{x}_i$ and $\mathbf{x}_j$ to denote two arbitrary data points or feature vectors, and $d_*(\mathbf{x}_i, \mathbf{x}_j)$ to denote the similarity or distance function between the two points.

- **Minkowski distance** - The Minkowski distance is a general distance function which is also a metric, and from which more specific distance measures are derived. It is defined as

$$d_p(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_p = \left(\sum_{k=1}^{d} (x_{ik} - x_{jk})^p \right)^{1/p},$$

  where p is real valued, acts as an adjusting factor, and is called the *norm level*.

- **Euclidean distance** - The Euclidean distance is the most popular distance function (not only in clustering) for continuous features. It is merely a special case of the Minkowski distance function ($p = 2$), and is defined as

$$d_2(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2 = \left(\sum_{k=1}^{d} (x_{ik} - x_{jk})^2 \right)^{1/2}.$$

  The Euclidean distance has an intuitive appeal as it measures the "physical" distance between points. It works well when the clusters are compact in shape and/or isolated. However, it has a tendency of letting the largest-scaled feature dominate the others. Some solutions to this problem include normalization of the features or other weighting schemes. In addition, the Euclidean distance has

a tendency to concentrate more and more around its expectation as dimensionality increases. In high dimensions, the Euclidean distances from a point to its nearest and farthest neighbors tend to be very similar and nearly meaningless. This problem if often referred to the *curse of dimensionality* and will further be discussed in chapter 5.

- **City-block (Manhattan) distance** - This distance is another instance of the Minkowski distance function ($p = 1$), and examines the absolute differences between coordinates of a pair of objects. It is defined as

$$d_1(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_1 = \sum_{k=1}^{d} |x_{ik} - x_{jk}|.$$

  In most cases, this distance measure yields results similar to the Euclidean distance. However, the effect of single large differences (outliers) is somewhat dampened since they are not squared.

- **Chebyshev distance** - The Chebyshev distance also called *maximum value distance* is another instance of the Minkowski distance ($p = \infty$). It is simply the maximum coordinate difference between a pair of objects regardless of any other differences, and is used in cases when one wants to define two objects as "different" or "dissimilar" if they are different on any one of the dimensions. Formally it is defined as

$$d_\infty(\mathbf{x}_i, \mathbf{x}_j) = \lim_{p \to \infty} \left(\sum_{k=1}^{d} (x_{ik} - x_{jk})^p \right)^{1/p} = \max_k |x_{ik} - x_{jk}|.$$

- **Mahalanobis distance** - The Mahalanobis distance uses the covariance matrix of the data, and is essentially Euclidean distance which has been rotated and

scaled. When the covariance matrix equals the identity matrix, the Mahalanobis distance reduces to the Euclidean distance. when the covariance matrix is diagonal, then the Mahalanobis distance is a normalized (by variances) Euclidean distance. It is mainly used when some of the features are linearly correlated or when some of the principle axes of a cluster have much larger variance than others. The Mahalanobis distance is defined as

$$d_M(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)'\Sigma^{-1}(\mathbf{x}_i - \mathbf{x}_j).$$

The next two similarity measures are used in cases where the behavior patterns of objects regardless of their magnitude need to be examined and compared. Rather than grouping objects with similar values that are physically or geometrically close to each other, objects are considered similar and grouped together if their behavior patterns are similar. These similarity measures are tightly related to correlation, and their main advantage over the previous measures are that they are scale invariant and do not depend on length. However, unlike the previous measures they are translation sensitive. Their main use is in the relatively new applications of DNA microarray analysis, collaborative filtering, and text mining. The concept of behavior pattern clustering will further be discussed in chapter 6.

- **Cosine distance (Angular separation)** - This similarity measure is simply the cosine of the angle between two vectors, and is defined as

  $$d_c(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i'\mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}.$$

  This angular separation measure ranges in $[-1, 1]$, similar to cosine, where higher values of angular separation (in absolute value) indicate that the two objects are "more" similar in their behavior patterns.

- **Pearson correlation** - This similarity measure essentially examines the linear correlation between two objects. It is a standardized angular separation that centers the coordinates to their mean value. Like the cosine distance is ranges in $[-1, 1]$, where higher values indicate the two objects are "more" similar in their behavior patterns. Formally it is defined as

$$d_r(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{k=1}^{d}(x_{ik} - \bar{x}_i)(x_{jk} - \bar{x}_j)}{\sqrt{\sum_{k=1}^{d}(x_{ik} - \bar{x}_i)^2}\sqrt{\sum_{k=1}^{d}(x_{jk} - \bar{x}_j)^2}}.$$

where $\bar{x}_i = 1/d \sum_{k=1}^{d} x_{ik}$ and $\bar{x}_j = 1/d \sum_{k=1}^{d} x_{jk}$ are the means or coordinate centers of objects $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively. The main problem with this measure is that it is not robust with respect to outliers, and may assign high similarity to a pair of dissimilar patterns. If two patterns have a common peak of valley at a single feature, the correlation will be dominated by this feature. Another problem is that it assumes an approximated Gaussian distribution of the points and may not be robust for non-Gaussian distributions. It can also be shown that the effectiveness or result of a clustering algorithm that uses Euclidian distance will be the same as an algorithm that uses Pearson correlation if *standardization* is applied to the data prior to the clustering. A standardization is a process that transforms each object $\mathbf{x}_i$ to another object $\mathbf{y}_i$ that has mean zero and variance one, i.e., for each feature k of object i,

$$y_{ik} = \frac{x_{ik} - \bar{x}_i}{\hat{\sigma}_i}$$

where $\hat{\sigma}_i = \sqrt{1/d \sum_{k=1}^{d}(x_{ik} - \bar{x}_i)^2}$ is the biased standard deviation of the values of object i. Given two standardized objects $\mathbf{y}_i$ and $\mathbf{y}_j$, one can easily show that

$$d_r(\mathbf{x}_i, \mathbf{x}_j) = d_r(\mathbf{y}_i, \mathbf{y}_j),$$

and that

$$d_2(\mathbf{y}_i, \mathbf{y}_j) = \sqrt{2d(1 - d_r(\mathbf{x}_i, \mathbf{x}_j))}.$$

Hence, if a pair of objects has larger (positive) correlation than another pair, then the Euclidian distance of that pair of objects will be smaller.

3.6 Cluster Models and Geometries

As mentioned, each clustering algorithm makes a very cardinal assumption (among others) either about the shape (geometry) or model of the clusters it is designed to identify.

The most frequently assumed, most intuitive, and somewhat trivial cluster geometry is the convex shaped or hyper-spherical cluster geometry. It can be described as a compact point cloud almost symmetrically distributed or centered around a representative point which is typically the cluster mean as depicted in Fig. 3.1. Most clustering algorithms are designed to identify this cluster shape as it captures the simple notion of points being similar if their feature values are similar, or in other words if they are geometrically close to each other. Another possibility is that these type of clusters come to capture the assumption that the cluster points are merely a small perturbation of a single representative point.

A slightly more complex cluster geometry is the hyper-ellipsoidal cluster geometry depicted in Fig. 3.2. In this case, the cluster points are still symmetrically distributed around a center point, but unlike spherical clusters the variances of the points along the principle axes of a cluster are not the same. These type of clusters are typically postulated when it is assumed that some of the cluster features are correlated. When clusters are well separated the simple Euclidian distance can be used to reveal these

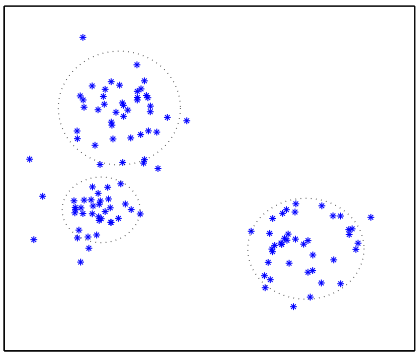


Figure 3.1: hyper-spherical clusters.

clusters, however in the case that the clusters are not well separated more elaborate distance measures such as the Mahalanobis distance might be necessary.

A large class of clustering algorithms are designed to identify arbitrary shaped clusters. By definition these clusters do not have a predefined shape, but are characterized as dense regions of points separated by low density regions, or from a graph theoretic point of view are characterized as connected components. Some of these type of clusters are depicted in Fig. 3.3. Algorithms designed to identify these type of clusters focus on density in prespecified volumes or point neighborhoods rather than on representative or central points as in the case of the hyper-spherical or hyper-ellipsoidal cluster shapes. The main problem with these type of clusters is their relative lack of interpretability.

Another type of cluster geometry that is often assumed is the so called "linear" cluster geometry, depicted in Fig. 3.4. These type of clusters are characterized as sets of points that follow a linear structure such as lines, hyperplanes, or the more general structure of linear manifolds - the subject of this thesis. This cluster geometry is

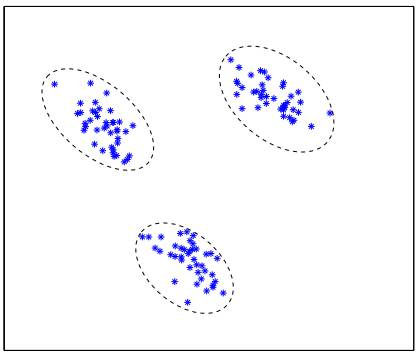


Figure 3.2: hyper-ellipsoidal clusters.

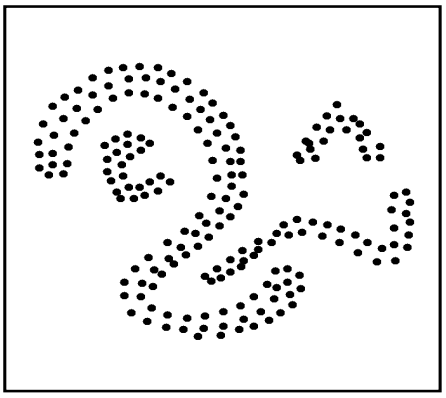


Figure 3.3: arbitrary shaped clusters.

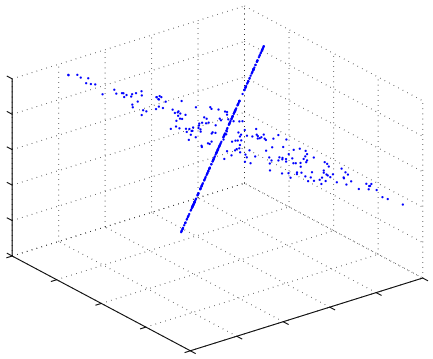


Figure 3.4: linear clusters.

typically assumed when it is suspected that a linear dependency or correlations exits among the features of the data. To the best of our knowledge there are no clustering algorithms that are specifically tailored to identify this type of cluster geometry. Existing methods that are able to detect these clusters either focus on density, which is not always adequate (points can be far and may not necessarily from dense regions, yet follow a linear structure), or rely on less intuitive and not always appropriate similarity measures such as the cosine or Pearson correlation measures.

A relatively new postulated cluster geometry is that of a non-linear manifold, a generalization of linear manifolds. Non-linear manifold clusters are characterized by sets of points that lie on smooth lower dimensional surfaces such as the one depicted in Fig. 3.5. Detection techniques for nonlinear manifold clusters are part of a new growing area of research called *manifold learning* which is beyond the scope of this thesis.

From a model based statistical point of view clusters are assumed to be generated by a "process" that can be described (modeled) by two parts, a deterministic and

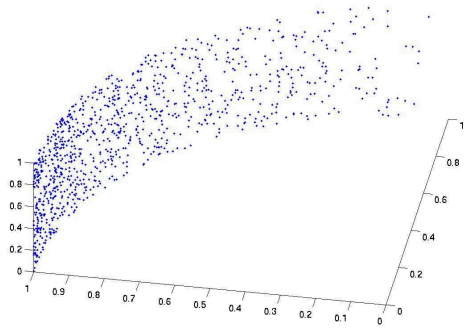


Figure 3.5: non-linear manifold cluster.

a stochastic. The deterministic part specifies the signal or "core" description of the cluster (e.g., the mean of the cluster or the linear manifold associated with it). The stochastic part specifies the degree to which the population of the cluster deviates from the deterministic part (e.g., cluster covariance or errors), and has a probability distribution associated with it. Several clustering methods take this approach and either cast the clustering problem to a problem of determining the model parameters, or tailor the clustering algorithm to detect the deterministic part (typically by a "model fitting technique"), taking into account the possibility of deviation from it as described by the stochastic part. The linear manifold clustering methods proposed in this thesis are members of the second approach.

The simplest and most popular cluster model that is often assumed and which captures the hyper-spherical or hyper-ellipsoidal cluster geometries considers the points of a cluster to be drawn from a Gaussian distribution characterized by a mean and a covariance matrix. In this case each point $\mathbf{x}$ belonging to a cluster can be modeled as

$$\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\epsilon}, \tag{3.6.1}$$

where $\boldsymbol{\mu}$ represents the mean of the cluster and the deterministic part of the model, and $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \Sigma)$ the deviation from the mean or the deterministic part of the model.

3.7 Clustering Schemes

In recent decades a multitude of clustering methods have been proposed. The main reason as mentioned earlier, is that there is no clustering method that is universally applicable to all clustering problems and applications. Unlike other reports, in this thesis we choose to classify the various clustering paradigms primarily based on the natural evolution of the clustering problems and applications they are designed to tackle. Then, within each of these primary classes we discuss subtypes or subclasses of clustering paradigms, each having its unique properties, strengths, and weaknesses.

Earlier or "classical" clustering methods typically dealt with smaller sets of features (lower dimensional data) and considered all the features of the data simultaneously relevant to each of the underlying clusters of the data. That is, each cluster identified by a clustering algorithm contained a subset of points each represented by the full set of features or dimensions underlying the data. Because of this, the name "full-dimensional" or "full-space" clustering has often been associated with these clustering methods. Hence, the first class of clustering methods discussed in this thesis (chapter 4) are the full dimensional clustering paradigms.

Due to recent technology advances in data collection many applications of clustering are now characterized by high dimensional data. These type of data sets pose new challenges which the full dimensional clustering methods are unable to address. The first challenge is that because of the relatively large feature set constituting the data, not all features may be relevant to each cluster, and the presence of non-information

carrying or irrelevant features has the potential to eliminate clustering tendency and mislead full dimensional clustering algorithms by masking clusters in noisy or irrelevant data. In other words, clusters are hidden and may be visible only in subsets or linear combinations of subsets of the features. The second challenge is the so called *'curse of dimensionality'*, which suggests that in high dimensions points tend to become equidistant from one another and therefore distance measures that utilize the full set of dimensions become increasingly meaningless. A recent advancement in the area of clustering was the introduction of "subspace clustering" methods designed to address these challenges. Consequently, our second class of clustering algorithms is the subspace methods discussed in chapter 5.

Lastly, in recent increasingly important applications of clustering such as gene expression microarray analysis, collaborative filtering, and web mining, object similarity is no longer measured by physical distance, but rather by the behavior patterns objects manifest or the magnitude of correlations they induce under a subset of features. Full-dimensional clustering methods and subspace clustering methods focus on grouping objects with similar values, and therefore are inadequate to capture coherent behavior patterns or correlations. Although effective in full-space clustering problems, the Pearson correlation similarity measure discussed in section 3.5 is not well suited for clustering similar behavior patterns that exist only in subspaces of the data, and has rarely been used in this context. The detection of coherent behavior patterns or correlations is a an important data mining task because they may reveal a dependency or some cause and effect relationship between the objects or features under consideration. The last class of clustering algorithm discussed in this thesis (chapter 6) are the pattern based or correlation clustering algorithms.

Chapter 4

Full-Dimensional Clustering Paradigms

4.1 Introduction

In recent decades a multitude of clustering methods have been proposed. Earlier or "classical" clustering methods typically dealt with smaller sets of features (lower dimensional data) and considered all the features of the data to be simultaneously relevant to each of the underlying clusters of the data. In other words, each cluster identified by a clustering algorithm contained a subset of points each represented by the full set of features or dimensions underlying the data. Because of this, the name "full-dimensional" or "full-space" clustering has often been associated with these clustering methods. Although somewhat limited, many clustering problems are still characterized by low-dimensionality data of which all the features are considered relevant. Moreover, the underlying concepts employed by full-space methods serve as a basis for the more sophisticated clustering approaches which localize the search for clusters in subspaces of the data.

Clustering algorithms are typically classified according the type of data on which they are applied, the similarity measure they employ, the clustering criteria they use,

and the underlying techniques used to group points and identify clusters. In this thesis we will focus on clustering algorithms that work on numerical attributes, as these are the most popular and widespread. Different reports have organized and classified clustering algorithms in different ways. Traditionally, the algorithms or paradigms were classified as belonging to one of two categories: the *partitioning* or *hierarchical* algorithms [5, 2]. However, over the years new methods have evolved, which cannot be classified as belonging to any of the above two categories. These new paradigms of clustering such as the *density-based*, *grid-based*, *fuzzy*, *model-based*, *artificial neural network-based*, and *graph-based*, have "earned" their own class, and along with the pervious two classes of clustering paradigms will be reviewed in the following.

4.2 Hierarchical Methods

Hierarchical methods create a hierarchy of nested clusters typically represented by a *dendrogram* - a tree which splits the data into recursively smaller and smaller clusters as depicted in Fig. 8.6. The tree can be created by an iterative process in a "bottom-up" fashion called the *agglomerative* approach, or in a "top-down" manner called the *divisive* approach, by merging or dividing clusters at each level of the tree. The agglomerative approach starts with each object in a separate cluster, and successively merges the appropriate objects/clusters according to some criteria (e.g., the distance between the two cluster centers) until all clusters are merged into one (top of the hierarchy) cluster, or until some termination condition is reached (such as the requested number of clusters K). The divisive approach on the other hand starts with all objects in one cluster, and in each successive iteration larger clusters are split into

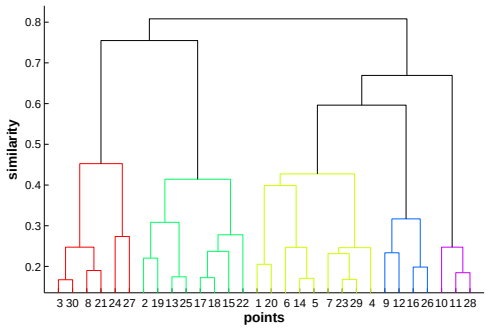


Figure 4.1: A sample dendrogram. The horizontal axis represents the points or clusters, and the vertical axis captures their similarity.

smaller more homogenous clusters according to some criteria, until eventually each cluster contains exactly one object or until some termination condition is reached.

The advantages of hierarchical methods are the ability to explore cluster structure or the data at different levels of granularity, and ease of handling any forms of similarity measures, making it applicable to a wide range of attribute types. Hierarchical algorithms do not typically require the number of clusters as an input parameter, but are very sensitive to a termination condition, which specifies when a merge or division operation should not be carried out. The main problem with hierarchical algorithms is the vagueness and difficulty associated with deriving the termination condition. Another disadvantage is that most hierarchical algorithms do not revisit constructed clusters for the purpose of their improvement.

Most hierarchical clustering methods are based on *Linkage Metrics*. Instead of operating on the regular object-attribute data structure, they operate on an $N \times N$ matrix of distances or similarities between points called *connectivity matrix*, from which linkage metrics are then constructed. Another disadvantage associated with

Hierarchical methods is the requirement of keeping these matrices in memory which for large data sets becomes problematic.

To merge or split clusters rather than individual points, the distance between individual points has to be generalized to the distance between clusters. Such a proximity measure is called a linkage metric, and the type of metric used significantly affects the algorithm as it reflects a particular concept of "closeness" and connectivity. Typical inter-cluster linkage metrics include the *single-link*, *average-link*, and *complete-link*. With the single-link method the distance between two clusters is the *minimum* of the distances between all pairs of points in the clusters, where each point comes from a different cluster. Similarly with the complete-link method the distance between two clusters is the *maximum* of the distances between all pairs of points in the clusters, and in the average-link case the distance between two clusters is the *average* distance between each pair of points. Most of the hierarchical algorithms are based on either the single-link or complete-link metrics. Algorithms based on the complete-link metric tend to produce tightly bound compact clusters, whereas the ones based on the single-link metric tend to produce straggly or elongated clusters [2]. Consequently, another advantage of hierarchical algorithms is their ability to deal with different geometries of clusters.

Among the earlier hierarchical algorithms are AGNES (AGglomerative NESting) and DIANA (DIvisia ANAlysis) [7]. AGNES (used in S-Plus) is a bottom-up method whereas DIANA a top-down. In both the number of desired clusters can be specified as a termination condition, and both can use the typical linkage metrics. Though simple, both often encounter difficulties pertaining to the selection of merge and split points. They will neither undo or revisit what was already done to improve cluster

quality, nor will they perform object swapping between clusters. Thus, the merge or split operations, if not well chosen at each step, may lead to low quality clusters. Both algorithms suffer from time complexity, which is $O(N^2)$, but despite that are very popular.

The BRICH (Balanced Iterative Reducing and Clustering using Hierarchies) [8] algorithm was introduced to overcome scalability issues associated with hierarchical algorithms. The main idea of BRICH is to compress the data objects into many small subclusters and then perform the clustering with these subclusters. The compression allows the clustering to be performed in main memory. Experiments show that BRICH scales linearly with the number of objects, but does not perform well when the clusters are not of spherical shape.

CURE (Clustering Using REpresentatives) [9] is an agglomerative method that employs two principles to obtain high quality clusters. First, instead of using a single object to represent a cluster, a fixed number of well-scattered objects are selected to represent each cluster. Second, the selected objects are shrunk towards their cluster centers. At each step, the two clusters with the closest pairs of representatives are merged. This allows CURE to adjust well to clusters of non-spherical shape. The shrinking of clusters helps dampen the effects of outliers. To handle large data sets, CURE employs a combination of random sampling and partitioning. A random sample is first partitioned, and each partition is partially clustered. The partial clusters are then clustered in a second pass to yield the desired clusters. As a stochastic algorithm the authors of CURE provide a complexity estimate defined in terms of sample sizes.

Similar to CURE, CHAMELEON [10] is an agglomerative algorithm that improves

clustering quality by using a more elaborate criteria for the merging operation. Two clusters are merged if their inter-connectivity and proximity of the points after the marge are similar to the inter-connectivity and proximity before the merging operation. CHAMELEON uses a weighted K-nearest neighbor connectivity graph, where each node represents an object, and an edge between two nodes exists if they are among the K-nearest neighbors of each other. The weight of the edge represents the closeness between two nodes. CHAMELEON operates in two stages. In the first, small tight clusters are built using a graph partitioning algorithm, and in the second stage the clusters are merged by an agglomerative process. It has been shown that CHAMELEON is more effective than CURE in discovering arbitrary-shaped clusters of varying density. It has complexity of $O(Nm + N \log m + m^2 \log m)$ where m is the number of sub-clusters built during the first stage.

Another famous hierarchical algorithm is ROCK [11] developed to work with categorical attributes.

4.3 Partitioning Algorithms

Partitioning clustering algorithms obtain a single partition of the data instead of a clustering structure, such as the dendrogram produced by hierarchical methods. The *partitioning* algorithms use a set of K representative objects to partition the data into K clusters. The representative objects are typically cluster centers such as in *K-means* algorithms, or objects located near the center such as in the *K-medoid* algorithms. The partitioning algorithms typically follow an iterative two step procedure which gradually improves cluster quality. In the first step the K representatives are selected by seeking to optimize some objective function such as minimizing the sum of squared

distances between the points and their representatives. In the second step each point is assigned to a cluster whose representative is closest. The main problem associated with these type of algorithms is that they expect the number of clusters as an input parameter, which requires prior knowledge usually not available. Another weakness of the partitioning-based algorithms is their inability to find arbitrary-shaped clusters. They are essentially limited to finding compact convex-shaped and well separated clusters. Partitioning methods have advantages in applications involving large data sets for which the computation of a dendrogram constructed by hierarchical methods is prohibitive. Typical partitioning methods have complexity which is linear with the size of the data set, i.e., $O(N)$. Another advantage of the these types of algorithms is their relative simplicity. Unlike traditional hierarchical methods, in which clusters are not revisited after being constructed, partitioning algorithms gradually improve clusters, which with appropriate data may result in high quality clusters.

K-means [12] is the simplest and most commonly used algorithm in this family of algorithms. The objective function used by the algorithm is the *squared error* $E(X)$ function defined as

$$E(X) = \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mathbf{m}_i\|^2. \tag{4.3.1}$$

It starts with a random initial partition and keeps reassigning objects to clusters based on the similarity between the object and the cluster centers $\mathbf{m}_i$, creating a new set of clusters. After the assignment the cluster centers are recomputed and the process is repeated until a convergence criterion is met (e.g., the squared error ceases to decrease, or their is no reassignment of objects from one cluster to another), or until a pre-specified number of iterations is reached. The main drawbacks with this algorithm is that it is sensitive to outliers since these can substantially influence

the cluster centers, and it heavily depends on the initial random partition that may lead it to converge to a local optimum. Another disadvantage is that because the cluster center is represented by its centroid (mean), the method works well only with numerical attributes. The K-medoids methods discussed next have no limitations on attribute types. The tremendous popularity of K-means has brought to life many other extensions and modifications. For example, Mahalanobis distance can be used to cover hyper-ellipsoidal clusters [13]. Maximum of intra-cluster variances, instead of the sum can serve as an objective function [14], and generalizations that incorporate categorical attributes are known [15]. The X-means algorithm [16] accelerates the iterative process in addition to searching for the best K in the process itself.

Unlike K-means, the K-medoids method uses a point–the most centrally located object in a cluster to be the cluster center. Because of this, the method is less sensitive to outliers, and is not limited to particular attribute types (e.g., numerical) like the K-means method. These advantages however, result in higher complexity. In the K-medoids method clusters are defined as subsets of points close to respective medoids, and the objective function is defined as the average dissimilarity between a point and its assigned medoid. The initializing state of the K-medoids method is the same as in the K-means method, that is, K objects are randomly selected as cluster centers. In the reassignment stage, K-medoids differs from K-means in that at most one cluster center will be changed in order to result in a decrease of the objective function. PAM (Partitioning Around Medoids) [7], one of the earlier versions of the K-medoids method, iterates through all the K centers and tries to replace each of them with one of the remaining $N - K$ objects if a decrease in the objective function is obtained. Because of this PAM does not scale well with large data sets

and its complexity is $O(K(N-K)^2)$. To deal with larger data sets, a sampling-based method, called CLARA (Clustering LARge Applications) was developed [7]. Instead of dealing with the whole data set, a small sample is selected. Medoids are chosen from this sample using PAM and the objective function is computed for the whole data set. To improve clustering quality multiple samples can be drawn and the best clustering returned. The complexity of CLARA is $O(Ks^2 + K(N-K))$ where s is the sample size. The effectiveness of CLARA heavily depends on the sample size. To improve the quality of the clustering and scalability of CLARA, CLARANS (Clustering Large Applications based upon RANdomized Search) [17] was developed. CLARANS tries to find a better center by picking one of the existing K centers and tries to replace it with a randomly chosen object from the remaining $N-K$ objects. If the process does not result in a decrease of the objective function, a local minima is assumed and the algorithm is restarted, where the number of restarts is specified by the user, and upon termination the best clustering is returned. CLARANS has been shown experimentally to be more effective than PAM and CLARA, however its complexity is still around $O(N^2)$.

4.4 Density-based Methods

Density-based methods characterize clusters as dense regions of objects separated by low density regions (noise). A cluster defined as a dense connected component grows in any direction that its density leads. Therefore, density-based methods are capable of discovering clusters of arbitrary geometries, and are protected against noise or outliers. They scale well with the size of the data. However, from a data description point of view they somewhat lack interpretability, and most do not scale well with

the dimensionality of the data.

DBSCAN (Density Based Spatial Clustering of Applications with Noise) [18], the first density based algorithm, judges the density in the vicinity of an object as sufficiently dense if the number of points within some distance (radius) ϵ of the object is grater than a certain threshold $MinPts$. If that is the case the object is called a *core object*. The algorithm grows regions with sufficiently high density into clusters. It does do so by checking the ϵ-neighborhood of each point in the data. If the ϵ-neighborhood of a point $\mathbf{x}$ contains more than $MinPts$, a cluster with $\mathbf{x}$ as its core object is created. Objects in the ϵ-neighborhood of $\mathbf{x}$ are then added to this new cluster. Core objects among the added members of the cluster will then go through the same process as $\mathbf{x}$ to grow the cluster. When no more core objects are found in the growing process, another core object will be selected from the data to grow another cluster. During the growing process if a core object that already belongs to another cluster is encountered the two clusters will be merged. The algorithm terminates when no new point can be added to any cluster. The complexity of DBSCAN is $O(N \log N)$. The main drawback with DBSCAN is that the clustering quality heavily depends on the input parameters ϵ and $MinPts$, and relies on the user's ability to select a good set of parameters. Without any guidance in setting these parameters a lot of time can be wasted on a *trial and error* approach. Moreover, different parts of the data may require the parameters to be set differently.

To overcome the problems associated with DBSCAN, OPTICS (Ordering Points To Identify the Clustering Structure) [19] was proposed. OPTICS takes the same input parameters as DBSCAN, however instead of producing a clustering based on one pair of parameters, OPTICS produces an ordering of the data points such that

the clustering for any lower values of ϵ and similar values of $MinPts$ can be visualized and computed easily. With each point OPTICS stores two additional fields, the core and reachability-distances. The *core-distance* of a point $\mathbf{x}$ is the smallest distance so that its neighborhood contains exactly $MinPts$ objects when it does not exceed ϵ, or undefined otherwise. The *reachability-distance* of $\mathbf{x}$ with respect to another object $\mathbf{y}$ is the smallest distance so that $\mathbf{x}$ is within the reach-neighborhood of $\mathbf{y}$, and $\mathbf{y}$ remains a core object with respect to the reach. If $\mathbf{y}$ is not a core object with respect to ϵ then the reach is undefined. Experimentally, OPTICS exhibits runtime roughly equal to 1.6 of DBSCAN's runtime.

Both DBSCAN and OPTICS rely on an indexing structure to process neighborhood queries efficiently. However the efficiency of these structures drops increasingly with the number of dimensions. To handle high dimensional data, DENCLUE (DENsity-based Clustering) [20] models the overall density of a point analytically as the sum of influence functions of the points around it. The underlying ideas of DENCLUE are:

1. The influence of each point can be formally modeled using a function called an *influence function*, which describes the impact of a data point within its neighborhood.

2. The overall density of the data can be modeled as the sum of the influence function of all the points.

3. Clusters can be determined by identifying *density attractors*, which are local maxima of the overall density function.

Let $f(\mathbf{x}, \mathbf{y})$ denote the influence function of object $\mathbf{y}$ on object $\mathbf{x}$. $f(\mathbf{x}, \mathbf{y})$ can be

an arbitrary function that takes as input a distance function $d(\mathbf{x}, \mathbf{y})$ (such as the Euclidian distance) and computes the influence. Two examples are the *square-wave* function

$$f_S(\mathbf{x}, \mathbf{y}) = \begin{cases} 0 & \text{if } d(\mathbf{x}, \mathbf{y}) > \sigma \\ 1 & \text{otherwise,} \end{cases}$$

and the *Gaussian* function

$$f_G(\mathbf{x}, \mathbf{y}) = \exp(-d(\mathbf{x}, \mathbf{y})^2/2\sigma^2),$$

where σ is an input parameter. The density function at an object $\mathbf{x}$ is defined as the sum of the influence functions of all the points, i.e.,

$$f_X(\mathbf{x}) = \sum_{\mathbf{y} \in X} f(\mathbf{x}, \mathbf{y}),$$

from which a gradient function $\nabla f_X(\mathbf{x})$ at $\mathbf{x}$ can be derived, which indicates the strength and direction where most of $\mathbf{x}$'s influence comes from. The density function is also used to locate *density attractors* which are the local maximas in the overall density function. $\mathbf{x}$ is said to be density attracted to $\mathbf{x}^*$ if there exists a set of points $\mathbf{x} = \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k = \mathbf{x}^*$ such that the gradient of $\mathbf{x}_{i-1}$ is in the direction of $\mathbf{x}_i$ for $0 < i < k$. A gradient ascent technique can then be used to determine the density attractor of each point. In addition to *center-defined* clusters, arbitrary-shape clusters are defined as sequences of points whose local densities are no less than a prespecified threshold ξ. The algorithm is stable with respect to outliers, and scales well. While no clustering algorithm can have complexity less than $O(N)$, the runtime of DENCLUE scales with N sublinearly! The explanation is that although all the points are fetched, the bulk of analysis during the clustering involves only points in highly populated areas.

Other representative density-based algorithms include: GDBSCAN[21] a generalization of DBSCAN and DBCLASD (Distribution Based Clustering of Large Spatial Databases) [22].

4.5 Grid-based Methods

As mentioned earlier, density-based methods do not scale well with the dimensionality of the data. To enhance the efficiency of density-based methods a new class of algorithms that use a grid data structure were developed. Grid-based methods quantize the space into a finite number of cells of the same (or varying) volume to form a grid structure, and measure the density of points within each cell. Dense cells which contain a certain number of points and which are close to each other are then combined to form clusters. The main advantage of this approach is the fast processing time, which is dependent only on the number of cells in each dimension of the quantized space. However, the quality of the clustering depends on the granularity level of the grid structure.

STING (STatistical INformation Grid) [23] is a multiresolution grid-based approach which uses statistical information stored in the grid cells to cluster the data. The gird structure used by STING is made of several layer (levels) organized in a hierarchical way, where each parent cell is divided into four cells at the next level. The information stored in each cell includes: count, mean, standard deviation, min, max, and the type of distribution that the attribute in the cell follows (normal, uniform, exponential, or none), where the last five are attribute dependent. Information stored in higher level cells can easily be computed from lower level cells. The information is accumulated starting from the bottom level layer and propagated to higher levels.

Using a density level input parameter STING clusters the data using a top-down approach that searches for regions with sufficient density. A layer in the grid structure that typically contains a small number of cells is selected to start the clustering. For each cell in a currently examined layer, a confidence interval is computed to determine if the cell is relevant to the clustering. The irrelevant cells are then refined to a finer resolution and the process is repeated, until the bottom level is reached. The relevant cells after being identified are connected to form clusters in a way similar to DBSCAN. STING passes through the data once to construct the grid structure. After that the complexity of clustering process depends on the number of grid layers and the number of cells at the lowest level, both typically much smaller than N, hence the overall complexity of STING is $O(N)$. The clustering quality of sting depends on the level of granularity of the lowest grid level. If it is too fine then processing time will increase, if it is too coarse then the quality of the clustering will be reduced. Defining the appropriate granularity is not straightforward. In addition STING is only able to detect clusters of isothetic shape, that is, the cluster boundaries are parallel to the axes of the coordinate system.

WaveCluster [24] is a multiresolution grid-based algorithm which applies a *wavelet transformation* on the original feature space to filter the data and find dense regions in the transformed space where clusters are more visible. A wavelet transform is a signal processing technique used to localize and filter signals. The hat shaped wavelet filters and emphasizes dense regions where points cluster, while less dense regions outside of the cluster boundaries can be suppressed. Hence, clusters can become more visible in the transformed space, and outliers eliminated. The multiresolution property of the algorithm also enables the detection of clusters at various levels of accuracy. The

complexity of WaveCluster is $O(N)$ and its implementation can be made parallel. Besides being efficient, and able to deal with outliers, it is also capable of detecting clusters of arbitrary shape. The authors of WaveCluster showed in experimental studies that it outperforms BIRCH, CLARANS, and DBSCAN both in terms of efficiency and cluster quality.

Other grid-based algorithms include: FC (Fractal Clustering) [25], GRIDCLUS [26], STING+ [27], and others such as CLIQUE [28] which are classified as subspace clustering algorithms and reviewed in chapter 5.

4.6 Fuzzy Clustering

The methods presented thus far generate "hard" or discrete clusterings, that is, each object is assigned to one cluster only. In many applications, like in real life, the clustering needs to reflect uncertainty. In gene expression clustering for example some genes may belong to multiple functional categories, and thus hard clustering may not always be adequate. Unlike hard clustering, "fuzzy" clustering allows objects to belong to multiple clusters by associating each object with each cluster by its degree of membership. These degrees or functions of membership typically range in $[0, 1]$, where larger values indicate higher confidence in the assignment of an object to a cluster.

The most prominent fuzzy clustering algorithm is Fuzzy C-Means (FCM) [29], an extension to the classic K-means algorithm. FCM uses an $N \times K$ membership matrix U, where each element in the matrix u_{ij} represents the degree of membership of object $\mathbf{x}_i$ to cluster C_j and ranges in $[0, 1]$. Similar to K-means the objective is to

minimize a "weighted" squared error function

$$E(X) = \sum_{j=1}^{K} \sum_{i=1}^{N} u_{ij}^{m} \|\mathbf{x}_i - \mathbf{c}_j\|^2, \tag{4.6.1}$$

where m, a weighting exponent is any real number greater than 1 and $\mathbf{c}_j$ is the centroid of cluster C_j. FCM like K-means starts with a random initial partition, in this case a random initialization of the matrix U, and clusters the data through an iterative optimization process similar to K-means by updating

$$\mathbf{c}_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} \mathbf{x}_i}{\sum_{i=1}^{N} u_{ij}^{m}} \tag{4.6.2}$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{K} \frac{\|\mathbf{x}_i - \mathbf{c}_j\|}{\|\mathbf{x}_i - \mathbf{c}_k\|}^{2/(m-1)}}, \tag{4.6.3}$$

until convergence. Even though FCM is considered better than K-means at avoiding local minima, it is still susceptible to the same problem.

A fuzzy c-shell algorithm and an adaptive variant for detecting circular and elliptical boundaries was presented in [30]. Another approach to fuzzy clustering is based on probabilistic models and discussed in the next section.

4.7 Model-based Methods

Model based methods also known as probabilistic methods assume that the patterns or objects are drawn from a mixture model of several probability distributions, each with its own set of parameters. Rather than performing classical clustering, the goal is typically to estimate the parameters of each model, whereupon point label or cluster assignment can be computed to generate the clustering. The estimation is typically done using an iterative optimization scheme similar to the scheme used by partitioning methods, and because of this model-based methods are traditionally classified as

partitioning methods. More specifically in the model based setting the clusters $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$ are considered classes. The main assumption is that each point $\mathbf{x}$ is generated by first randomly picking a model or class C_i with probability $P(C_i)$ and then drawing the point from the corresponding probability distribution $p(\mathbf{x}|C_i)$. In this setting it is also assumed that each point is associated with a corresponding hidden class or cluster label, and belongs to only one cluster. Given the above and using Bayes' rule, the probability of assignment of a point to the i-th class,

$$P(C_i|\mathbf{x}) = \frac{p(\mathbf{x}|C_i)P(C_i)}{\sum_{j=1}^{K} p(\mathbf{x}|C_j)P(C_j)} \tag{4.7.1}$$

can be computed, and used to cluster the data either in a "hard" or "fuzzy" way. In addition, by combining the effects of the different distributions at $\mathbf{x}$, the mixture model probability density function for $\mathbf{x}$ can be computed as

$$p(\mathbf{x}) = \sum_{i=1}^{K} p(\mathbf{x}|C_i)P(C_i). \tag{4.7.2}$$

Using eq. (4.7.2) the *likelihood* of the data

$$L(X|C) = \prod_{i=1}^{N} p(\mathbf{x}) = \prod_{i=1}^{N} \sum_{j=1}^{K} p(\mathbf{x}|C_j)P(C_j) \tag{4.7.3}$$

can be computed, whereupon *maximum likelihood estimation* can then be used to estimate the parameters of the mixture of models. In maximum likelihood estimation the *log-likelihood* $\log(L(X|C))$ typically serves as the *objective function*, and the maximum likelihood parameter estimates are obtained when this function reaches a maxima.

The Expectation-Maximization (EM) algorithm [31] is a general method of finding the maximum likelihood estimate of the parameters of an underlying distribution from a given data set when the data is incomplete or has missing values. The EM has

two main applications. The first is when the data has missing values due to problems or limitations with the observation process. The other, which applies to the case of mixture models and which is more common in pattern recognition problems, is when the optimization of the likelihood function is analytically intractable but can be simplified by assuming the existence of additional but *missing* or *hidden* variables, such as class labels in our case. The EM algorithm, similar to other optimization schemes, is an iterative optimization technique which gradually improves the parameter estimates. It can be viewed in many different ways, where one of the most insightful being in terms of lower bound optimization [32]. Unlike gradient ascent [5] which makes a local linear approximation to the objective function or Newton methods [33] which make quadratic approximations at each iteration and then take some *uphill* step, EM makes a local approximation which is a lower bound approximation of the objective function. Choosing a new guess to maximize the lower bound will always be an improvement over the previous guess, unless the gradient there was zero. The main difficulty with gradient ascent or Newton methods which is not present in the EM scheme is that we do not know in advance how good the linear or quadratic approximations are, neither do we know in advance how big of an uphill step must be taken. An illustration of this concept is presented in Fig. 4.2.

Similar to the K-means algorithm, the EM is a two-step iterative process. Starting with an initial guess of the parameters that need to be estimated. Each iteration consists of two steps. The first, an Expectation (E) step for computing a local lower bound approximation to the objective function (log-likelihood), and maximizing it with respect to the distribution of the unobserved data. This step is equivalent to finding the distribution of the unobserved or missing variables given the observed data

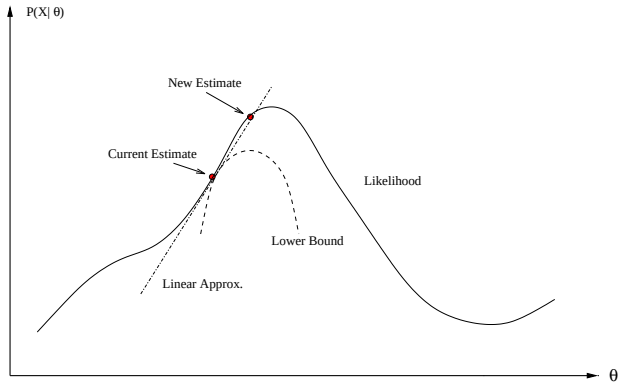


Figure 4.2: linear vs. lower bound approximation for maximum likelihood. θ is the set of parameters that need to be estimated, and $P(X|\theta)$ the likelihood of the data X given the parameters.

and the current parameter estimates, that is estimating $P(C_i|\mathbf{x})$ for each class/cluster with which a fuzzy assignment can be made. The second step is a Maximization (M) step which maximizes the lower bound with respect to the parameters of the underlying distribution given the distribution of the missing data found in the E-step. That is, estimating the parameters of the distribution $P(\mathbf{x}|C_i)$ for each class/cluster. These two steps are repeated until convergence of the parameter estimates is reached, or until a local maximum is found. Most of the work in this area and clustering in particular assumes that the individual components of the mixture distribution are Gaussian, i.e., that the clusters are ellipsoidal in shape. In the mixture of Gaussians case the parameters $P(C_i)$, $\boldsymbol{\mu}_i$ (mean), and Σ_i (covariance matrix) need to be estimated for each cluster, and each iteration of the EM algorithm consists of restimating the parameters as follows,

$$p(C_i) = \frac{1}{n}\sum_{j=1}^{N} p(C_j|\mathbf{x}_j) \tag{4.7.4}$$

$$\boldsymbol{\mu}_i = \frac{\sum_{j=1}^{N} \mathbf{x}_i p(C_i|\mathbf{x}_j)}{\sum_{j=1}^{N} p(C_i|\mathbf{x}_j)} \tag{4.7.5}$$

$$\boldsymbol{\Sigma}_i = \frac{\sum_{j=1}^{N} p(C_i|\mathbf{x}_j)(\mathbf{x}_j - \boldsymbol{\mu}_i)(\mathbf{x}_j - \boldsymbol{\mu}_i)'}{\sum_{j=1}^{N} p(C_i|\mathbf{x}_j)}, \tag{4.7.6}$$

essentially the maximum likelihood estimates of the parameters of a single Gaussian, except that each point is now weighted by the probability that it belongs to the i-th cluster.

Model-based clustering has some clear advantages over other clustering paradigms. Because it has a solid probabilistic foundation the clustering results are easily interpretable and can easily be extended to build meaningful classifiers. Problems like determining the number of clusters K or the best clustering become more tractable and can be addressed within a probabilistic setting. The method can easily be modified to handle more complex cluster structures. However, like many other iterative optimization techniques model-based clustering and the EM algorithm in particular is a local technique and as so is just as susceptible as other techniques to local optima. The rate of convergence is typically good during the first few iterations, but can become very slow as it approaches the local optima. Generally the EM algorithm works best when the fraction of missing information is small and the dimensionality of the data is not too large.

Other model-based algorithms include AutoClass [34], which covers a wide variety of distributions and extends the search for different models and different K using a Bayesian framework. SNOB [35] which uses the MML (minimum message length) principle [36]. The algorithm MCLUST [37] uses Gaussian models of ellipsoids of different volumes, shapes, and orientations. It is a software package for hierarchical mixture model clustering, and discriminant analysis, which uses the BIC (Bayesian

Information Criterion) [5] for estimation of goodness of fit.

4.8 Artificial Neural Network Clustering

Artificial Neural Networks (ANNs) are motivated by biological neural networks. Each pattern is represented by a node (neuron) in the network and each node is associated with a weight (vector). The networks are trained by evaluating the similarity between the input patterns presented to the network and the neurons. For example, if the Euclidian distance is used as a measure of the similarity between input vectors and network weights, then an unsupervised network adapts its weights to become more like the frequently occurring input vectors. The basic form of learning employed in unsupervised neural networks is called competitive learning or "winner-take-all" [38]. The neurons compete among each other to be the one that is triggered or adapted first by their similarity to the input vector. Typically, all the neurons in the network are identical except that they are initialized to have randomly distributed weights. ANNs can be used to perform clustering by grouping similar neurons. The architecture of ANNs used to perform clustering is simple: they are single layered, patterns are presented at the input and are associated with output nodes, and the weights attached to the neurons are iteratively changed in the learning phase until a termination condition is reached.

The most renowned example of an ANN used for clustering is Kohonen's Learning Vector Quantization (LVQ) and the Self Organizing Map (SOM) [39]. SOM is a subtype of artificial neural networks. It is trained using unsupervised learning to produce a low dimensional representation of the training samples while preserving the topological properties of the input space. This makes SOM a tool for visualizing

high-dimensional data in much lower-dimensional spaces. The process of reducing the dimensionality of vectors is essentially a data compression technique known as vector quantization, hence the name LVQ. SOM consists of a (typically) one or two dimensional array of neurons. When an input pattern is presented in parallel to all the neurons, they compete against each other to represent the pattern. The neuron whose vector of weights is closest to the input pattern wins the competition. The winner and the neurons close to it (its neighbors) are then updated by moving their weight vectors closer to the input pattern. An illustration of this concept is depicted in Fig. 4.3. Because neurons near the winner are also updated, as training progresses neurons that are neighbors tend to represent similar patterns, while neurons far from each other in the map represent dissimilar patterns. If there are clusters, then the points within a cluster will tend to activate the same neuron, while points from different clusters will be represented by separate neurons. The more dissimilar the clusters the further apart they will be mapped in the output layer. The neuron with weight vector most similar to the input vector $\mathbf{x}$ is called the Best Matching Unit (BMU). The weights of the BMU and neurons close to it in the SOM are adjusted towards the input vector. The magnitude of the change decreases with time and is smaller for neurons less similar from the BMU. The update formula for a neuron with weight vector W_t is of the form

$$W_{t+1} = W_t + \Theta_t \alpha_t (\mathbf{x} - W_t)$$

where t represents the time-step, W_t the old weight, W_{t+1} the new weight, α_t a monotonically decreasing coefficient called the learning rate, and Θ_t a neighborhood function representing the amount of influence a neuron's distance from the BMU has on its learning. In the simplest form it is one for all neurons close enough to

BMU and zero for others, but a Gaussian function is commonly used. Regardless of the functional form, the neighborhood function shrinks with time. At the beginning the neighborhood is broad, and the self-organizing map takes place on the global scale. When the neighborhood has shrunk to just a couple of neurons the weights are converging to local estimates. The learning rate is typically calculated by an exponential decay function of the form

$$\alpha_t = \alpha_0 \exp(-t/\lambda).$$

Basically, the update formula says that the new adjusted weight for a neuron is equal to the old weight plus a fraction of the difference between the old weight and the input vector. SOM can be interpreted in several ways. Because in the training phase weights of the whole neighborhood are moved in the same direction, similar items tend to excite adjacent neurons. Therefore, SOM forms a semantic map where similar samples are mapped close together and dissimilar far apart. The other way is to think of the neuron weights as pointers to the input space. They form a discrete approximation of the distribution of training samples. More neurons point to regions with high training sample concentration and fewer where the samples are scarce.

Despite its appealing properties, SOM like many other iterative methods may generate sub-optimal clusterings, specially if the initial weights are not selected properly. One way of choosing the initial weights is to first perform PCA (principle component analysis) on the whole data and then perturbate or sample from the first few eigenvectors. Moreover, its convergence is governed by various parameters, like the learning rate and neighborhood function, which result in different clustering for different values or functions. Further, the number of neurons is kept constant which can limit the number of clusters produced.

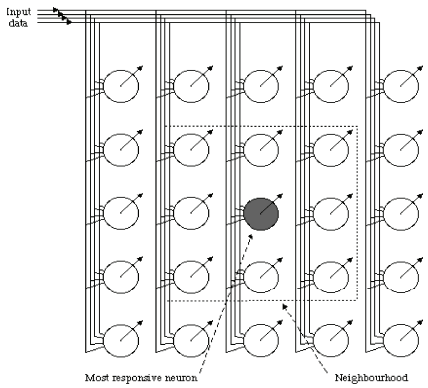


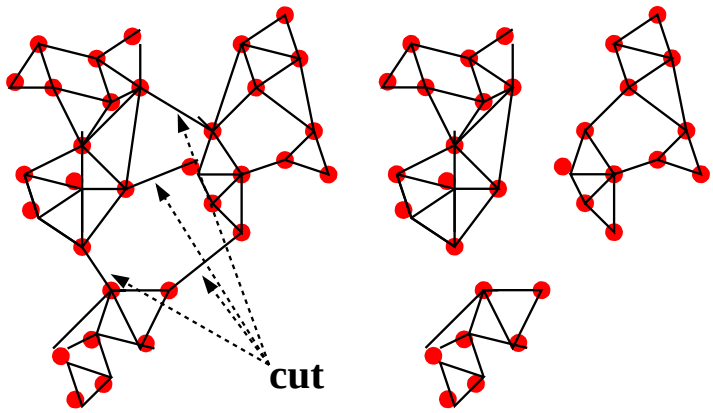
Figure 4.3: A two-dimensional self-organizing map.

4.9 Graph Clustering

The problem of clustering a set of points can be cast into a graph partitioning problem. The objects to be clustered correspond to the set of vertices V of the graph, and the weighted edge set E relate to the similarity between the objects. In graph based clustering the general goal is either to find a *minimum cut* or *maximal cliques* depending on the approach. A set of edges whose removal partitions a graph into K pair-wise disjoint sub-graphs, is called an edge separator or *cut*, as illustarted in Fig. 4.4. The objective is to find such a cut with a minimum sum of edge weights. Most methods and analytic solutions to the problem consider the simpler problem of bi-partitioning the graph, i.e. partitioning into two disjoint sets A and B, which in graph theoretic language is defined as

$$cut(A, B) = \sum_{v \in A, u \in B} w(v, u),$$

where $w(v, u)$ is the weight between vertices v and u. An algorithm to the find such a cut can then be applied recursively to find K partitions or clusters. Finding the



(a) before partitioning (b) after partitioning

Figure 4.4: Graph clustering by cuts.

cut of a graph is a well studied problem and efficient algorithms for some versions of it are known. One such method is the Kernighan-Lin (KL) algorithm [40]. The KL $O(N^3)$ is based on iteratively conducting best sequences of swaps involving all vertices. The problem with the cut as defined is that it favors cutting small sets of isolated vertices in the graph. Ideally, while striving for the minimum cut objective, the number of objects in each partition should be approximately equal. To avoid this bias of partitioning small sets of objects various *normalized cuts* were defined, one of which is [41]

$$Ncut(A,B) = \frac{cut(A,B)}{asso(A,V)} + \frac{cut(A,B)}{asso(B,V)},$$

where $asso(A,V) = \sum_{u \in A, t \in V} w(u,t)$, (and similarly $asso(B,V)$) is the total connection of vertices in A (B) to all the graph.

CLICK (Cluster Identification via Connectivity Kernels) [42] is a graph based algorithm designed to find clusters in gene expression data. It makes an assumption that the similarity values between elements are normally distributed, and that the

weights attached to edges represent the probability that two vertices are in the same cluster. The clustering process iteratively finds the minimum cut in the graph and recursively splits the data set into a set of connected components from the minimum cut.

CAST (Cluster Affinity Search Technique) [43] is another graph based algorithm also designed for gene expression data which is based on the idea of a *corrupted clique graph* model. Under this model the data is assumed to come an underlying cluster structure which is contaminated with random error caused by the complex process of gene expression measurement. It is assumed that the true clusters can be represented by a clique graph and the goal is to identify the cliques (clusters) from the corrupted version.

A promising alternative that has recently emerged is the use of *spectral methods* for clustering. In these methods the min cut problem is cast into a (generalized) eigenvalue problem, where the eigenvectors of the weighted adjacency matrix or the Laplacian [44] derived from it are used to partition the graph [45]. Spectral algorithms have been applied successfully in many domains including computer vision and VLSI design. But despite their success, different authors disagree on which eigenvectors to use and how to derive the clusters from them. Perona and Freeman [46] for example suggested a clustering algorithm based on thresholding the first eigenvector of the affinity (adjacency) matrix. Shi and Malik [41] use the second smallest generalized eigenvector of a normalized affinity matrix to cluster the data.

In addition to their simplicity and theoretical foundation, one of the main advantages of the spectral methods is that they are able to identify clusters of arbitrary shape. However, like other methods they do not guarantee an optimal solution.

Chapter 5

Subspace Clustering

5.1 Introduction

Due to recent technology advances in data collection many applications of clustering such as DNA microarray analysis in bioinformatics, document clustering of web pages, image segmentation in computer vision, and recommendation or collaborative filtering systems in E-commerce, are now characterized by high dimensional data which poses two challenges. First, very often not all features of the data are relevant to the clustering, and the presence of non-information carrying or irrelevant noisy features has the potential to eliminate clustering tendency and mislead the clustering algorithm by masking clusters in noisy or irrelevant data. In other words, clusters are hidden and may be visible only in subsets or linear combinations of subsets of the features. Second, is the so called *'curse of dimensionality'*, which suggests that the sparsity of the data increases exponentially with the dimensionality of the input space given a constant amount of data, with points tending to become equidistant from one another at a certain high dimension [47]. This in turn implies that learning structure in high dimensional spaces by distance measures that utilize the full set of dimensions becomes increasingly difficult.

Traditional clustering algorithms such as K-means, DBSCAN, and others reviewed in chapter 4 are "full-dimensional" in the sense that they give equal relevance or importance to all the dimensions of the data. In other words, they assume that all the attributes or features of the data are simultaneously information carrying for each of the clusters that may exist in the data. For this reason full-dimensional clustering algorithms are likely to fail when applied to high dimensional data, or data where not all attributes are relevant to each cluster.

Initial attempts to tackle this problem involved different types of dimensionality reduction techniques such as *feature selection* and *feature transformation* techniques as preprocessing steps whereupon full-dimensional clustering was performed in the reduced space. Feature selection attempts to discover the most revelent attributes of the data to reveal objects that are similar only in a subset of attributes. Feature transformation techniques on the other hand, such as Principle Component Analysis (PCA) transform the original space into a lower dimensional space in an attempt to summarize the data. While effective in reducing the dimensionality of the data, these methods are limited in that they can only be applied to the data set as a whole, and therefore are only capable of detecting structure which is expressed in the data as a whole. Feature transformation techniques generally preserve the relative distance between objects and are therefore ineffective when there are a large number of irrelevant dimensions that mask the clusters in noise. They do not actually remove any of the original features and thus information from the irrelevant features is preserved. In addition, the new features obtained after the transformation are linear combinations of the original features and may be very difficult to interpret in the context of the domain, making the clustering results less useful. Feature transformation techniques

are best suited to data sets where most of the dimensions are relevant to the clustering task, but many are highly correlated or redundant. Feature selection techniques on the other hand, have difficulties when clusters exist in different subspaces (subsets of features) of the data. Since different clusters may exist in different subspaces, removing dimensions globally is likely to incur a loss of crucial information. Thus, only methods that localize the search for relevant features cluster by cluster or first remove the irrelevant features and focus only on the relevant dimensions cluster by cluster are likely to succeed. Figs. 5.1-5.4 illustrate the concept of subspace clusters and the problems associated with the traditional techniques used to identify them. Fig. 5.1 shows a data set consisting of three clusters each existing in a different 2-dimensional subspace, where the third dimension is noise or an irrelevant dimension. In the full space the points are relatively sparse showing no apparent clustering tendency, which might have been captured by full-space clustering algorithms. However, Fig. 5.2 shows three different projections of the same data onto 2-dimensional subspaces. In each projection a clustering tendency of one of the clusters is visible, and can therefore be identified by methods that localize the search for clusters in subspaces. Feature selection techniques would have likely chosen to eliminate one of the dimensions. But since each of the three clusters localizes to a different set of dimensions, only one of the clusters would have been identified, and therefore feature selection would have failed to aid in revealing all three clusters. Fig. 5.3 shows a projection of the same data onto the two leading principle components (eigenvectors) of the data obtained by PCA. Although some clustering tendency of one of the clusters may be seen, PCA is dominated by some of the irrelevant noisy dimensions, and therefore fails to reveal

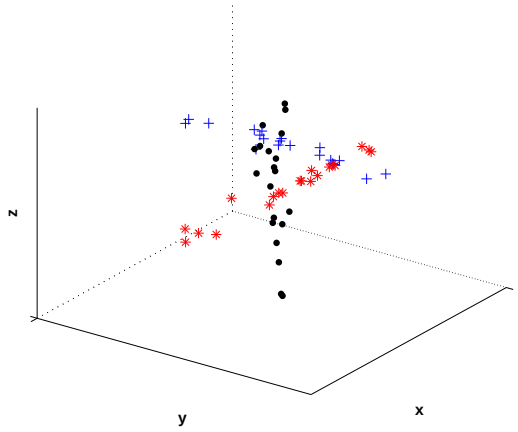
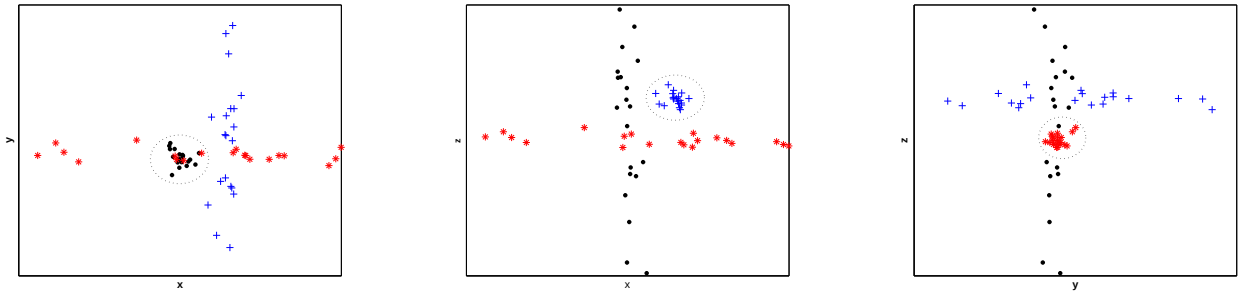


Figure 5.1: Three subspace clusters in the full space.

the structure of the data appropriately. That is, only one of the clusters can identified after reducing the dimensionality of the data using PCA. Fig. 5.4 illustrates one aspect (sparsity) of the curse of dimensionality. The figure shows that as dimensions are added to the data points, that might have been considered dense (in unit cells) in lower dimensions, they get stretched out and become sparser as in higher dimensions. Thus, eliminating clustering tendency.

5.2 The Subspace Clustering Paradigm

An important advance in the area of clustering was the introduction of *subspace clustering* as a paradigm of clustering applicable to high dimensional data, which is supposed to maintain cluster quality and efficiency. Subspace clustering is considered an extension to traditional clustering and feature selection techniques [48], in that it attempts to find different clusters embedded in different subspaces (subsets of features) of the same data set. In the subspace clustering paradigm a subspace



(a) x-y projection (b) x-z projection (c) y-z projection

Figure 5.2: 2D projections of the data from Fig. 5.1.

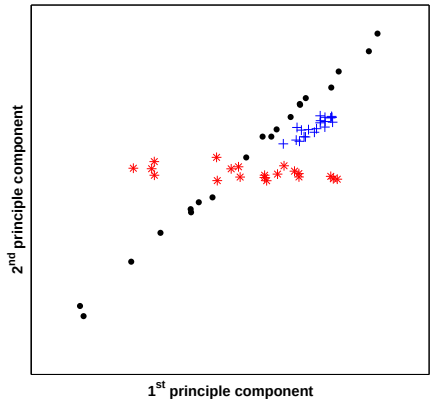
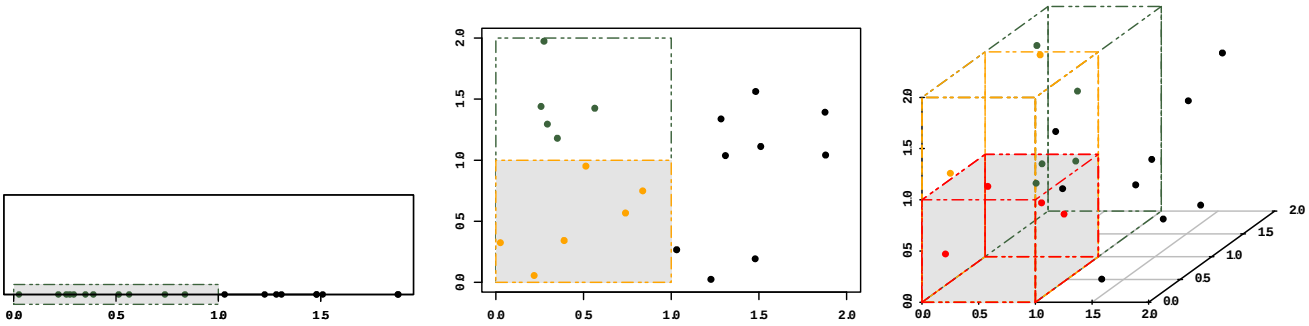


Figure 5.3: Projection of the data from Fig. 5.1. onto the two leading principle components.



(a) 11 points in a 1D unit cell (b) 6 points in a 2D unit cell (c) 4 points in a 3D unit cell

Figure 5.4: The curse of dimensionality. As dimensions are added points become sparser eliminating clustering tendency.

cluster consists of a subset of points and a corresponding subset of attributes, such that these points form a dense region in a subspace defined by the set of corresponding attributes. Unlike feature selection or transformation, subspace clustering methods localize their search for clusters in subspaces of the data. Most subspace clustering methods are restricted to finding clusters in subspaces spanned by some subset of the original measurement features, that is, *axis-parallel subspaces*, as those depicted in Fig. 5.1. In this context the term subspace is somewhat misleading since an axis-parallel subspace is only a special case of a subspace. The main reason for this restriction is, as with feature transformation techniques, the interpretability of the clustering results. However, examination of real data often shows that points tend to get aligned along arbitrarily oriented subspaces. One of the main drawbacks of subspace clustering algorithms is the inability to identify clusters embedded in arbitrarily oriented subspaces. At the expense of being less truthful to the data these algorithms are essentially trading off accuracy with interpretability. One of the main advantages of the linear manifold clustering paradigm is that it is able to address

this problem. This is because an arbitrary oriented subspace is merely an instance of a linear manifold, and therefore algorithms designed to identify linear manifold clusters will be able to identify clusters embedded in arbitrary oriented subspaces as a byproduct.

A naive approach to subspace clustering is to search through all possible subspaces for clusters. Needless to say that this approach is computationally intractable as the number of subspaces is exponential in the dimension of the data, and just enumerating the subspaces is an intractable problem. Like other clustering paradigms the subspace clustering paradigm relies on search heuristics to make the problem manageable. These search techniques or strategies determine the characteristics of the subspace clustering method. According to Parsons et al. [48] subspace clustering algorithms are primarily classified into two groups depending on the direction of search they employ. In the first group are the *bottom-up* algorithms that search for clusters from lower to higher dimensionality subspaces, and in the second group are the *top-down* algorithms that search for subspace clusters by going in the opposite direction, i.e., starting in higher dimensional subspaces and progressing to lower. Both methods are essentially *step-wise* methods (a term used to describe feature selection methods), they locate a cluster in an initial set of dimensions and depending on the direction of search, refine the cluster by either adding relevant dimensions or pealing off irrelevant dimensions.

5.3 Bottom-Up Methods

The *bottom-up* algorithms proceed from lower to higher dimensions by exploiting the *downward closure* property of density to prune the search space, similar to the

downward closure property used by the APRIORI algorithm in mining association rules [49]. The downward closure property means that if a collection of points X forms a cluster in a k-dimensional space, then X is also part of a cluster in all $(k-1)$-dimensional projections of this space. Hence, using the contrapositive a pruning rule can be devised-given a potential k-dimensional dense region: if we find that one of its $(k-1)$-dimensional projections is not dense we can conclude that the k-dimensional region is not dense, and prune it from the search space. Most of the bottom-up methods rely on grid-based and density-based clustering methods. Some use a static grid and some a dynamic one which adapts to the data. They start the search for dense grid units in lower dimensional subspaces and extend them to higher dimensions until the units are no longer dense. Adjacent units are then combined to form clusters. The nature of these methods usually lead to overlapping clusters which for some applications is an advantage. They heavily depend on proper tuning of the gird size and density threshold parameters, especially when set to the same values across all dimensions.

CLIQUE (CLustering In QUEst)[28] is probably the most famous and one of the first subspace clustering algorithms. Like the bottom-up algorithms it combines grid and density based approaches to cluster the data. In CLIQUE the space is partitioned along each dimension into non-overlapping rectangular units of fixed width. A unit is said to be *dense* if the fraction of points contained in it exceed a density input threshold, and a cluster is defined as a maximal set of *connected dense units*. CLIQUE uses the downward closure property of density to find clusterable spaces. Dense subspaces are sorted by *coverage*, defined as the fraction of points covered by the dense units in the subspace. Subspaces with largest coverage are kept and the

remaining pruned. CLIQUE finds adjacent dense units in each of the selected subspaces using a depth first search approach. Clusters are then formed by combining these units using a greedy growth scheme. The clustering results are described by DNF (disjunctive normal form) expressions. CLIQUE finds clusters in subspaces of the highest dimensionality. It is also able to find clusters of arbitrary shape in any number of dimensions. It scales well with the size and dimensionality of the data, but does not scale well with the dimensionality of the subspaces in which clusters are embedded. In fact, its complexity is exponential with respect to the dimensionality of the subspaces in which clusters exist, and because of that it is practical only when the clusters are assumed to be embedded in lower dimensionality subspaces. Moreover like many other subspace clustering algorithms it is restricted to finding axis-parallel subspaces, and heavily depends on proper tuning its input parameters.

The algorithm MAFIA (Merging of Adaptive Finite Intervals) [50] is an improvement to CLIQUE that uses adaptive data driven rather than static grids. MAFIA initially creates a histogram to determine the minimum number of bins for each dimension, and then combines bins of similar densities in each dimension to form larger cells. In this manner fewer potential dense cells are generated and cluster boundaries are captured more accurately. Once the bins have been defined, MAFIA proceeds much like CLIQUE exploiting the downward closure property of density. In addition to requiring the inputs that CLIQUE takes, MAFIA also requires the input of a threshold for merging adjacent windows, which are bins in the histogram. Adjacent windows within the specified threshold are merged to form larger windows. The formation of the adaptive grid depends on these windows. MAFIA also uses parallelism to improve efficiency. Like CLIQUE it is able to find clusters of arbitrary shape, and

uses DNF to describe the clusters. Also like CLIQUE it scales linearly with the size of the data, and better then CLIQUE, with the dimensionality of the data. However, just like CLIQUE, it is exponential with respect to the dimensionality of the subspaces in which clusters exist. In addition, similar to other methods it is confined to finding clusters in axis-parallel subspaces.

Other bottom-up algorithms include: ENCLUS (ENtropy based CLUStering) [51], which based on the CLIQUE algorithm but uses entropy instead of density or coverage to qualify subspace clusters. The algorithm is based on the observation that subspaces with clusters have lower entropy than those without. ENCLUS has the same complexity as CLIQUE and suffers from the same drawbacks, but is supposed to generate clusters of higher quality. CLTREE (CLustering based on decision Trees) [52] uses a decision tree to partition each dimension and separating hyperplanes to separate low and high density regions. Virtual noise points are added to the data to perform the clustering. One of its main advantages similar to the full-space hierarchical algorithms is the ability of the users to browse the tree and refine the clustering results. CLTREE uses different parameters than the other algorithms and has complexity that is quadratic in the size of the data due to the tree building process.

5.4 Top-Down Methods

Top-down subspace clustering algorithms utilize an iterative improvement approach similar to the full-space partitioning algorithms. They usually start with an initial approximation of the clusters in the full space with equally weighted dimensions. Then at each iteration the weights or scores for each dimension are updated and used to regenerate improved or refined clusters by peeling off from them dimensions. The

main properties of the top-down algorithms is that due to their nature they create discrete clusters, usually of convex shape and do not allow for the possibility of overlapping clusters which in some applications is required. They also expect the number of clusters and the dimensionality of the subspaces as input parameters which are hard to determine in advance. Requiring the subspace dimensionality as an input parameter also causes the clusters to be found in subspaces of the same dimensionality, which happens only in rare cases. They require computationally expensive clustering techniques and many use sampling to improve the performance. Those that use sampling usually require an extra sample size input parameter which effects the clustering quality.

PROCLUS (PROjected CLUStering) [53] the first algorithm in this class is a subspace clustering extension to the approach used by CLARANS (a K-medoids algorithm reviewed in section 4.3). PROCLUS samples the data to select a set of K medoids and then iteratively improves the clustering. It proceeds in three phases: initialization, iteration, and cluster refinement. The objective of the initialization phase is to select a small superset of the best K-medoids, accomplished by sampling and a greedy technique. The objective of the iterative phase is to identify and gradually improve a set of K-medoids, each with respect to a reduced set of dimensions. The average distance along each dimension from the points in the locality of a medoid is computed. The reduced set of dimensions is then determined by selecting the dimensions whose average distance is smallest relative to statistical expectation. The total number of dimensions associated to medoids is subject to the restriction that it must be $K \cdot l$, where l is an input parameter that selects the average dimensionality of each cluster. After the subspaces have been identified, average Manhattan distance is used

to assign points to medoids. Finally in the refinement phase new dimensions are computed for each medoid based on the clusters and not the locality of the medoids. Like other partitional methods PROCLUS is biased towards finding discrete clusters that are spherical in shape and in axis-parallel subspaces. Since the average dimensionality of the subspaces in which clusters are assumed to exist is an input parameter, the clusters found by PROCLUS are of similar dimensionality. PROCLUS is slightly faster than CLIQUE due to the use of sampling, but is very sensitive to the values of the input parameters.

ORCLUS (arbitrarily ORiented projected CLUSter generation) [54] is an extension to PROCLUS that is able to find clusters in arbitrary oriented and not only axis-parallel subspaces. ORCLUS is an outcome of the realization that in most real life applications clusters tend to be embedded in arbitrarily oriented subspaces and not axis-parallel ones, in other words, that real life data tends to contain clusters that induce correlations between different subsets of attributes of the data. In a sense, among all clustering algorithms that we are aware about to this day, whether they are full-space or subspace clustering algorithms, ORCLUS is the closest relative to the linear manifold paradigm of clustering. The main difference which will be become clearer later on, is that rather than focusing on the linear manifold itself, ORCLUS focuses on the dimensions or vectors spanning the space orthogonal to the manifold, the space which according to the linear manifold cluster model (discussed in chapter 7) contains the random perturbation off the manifold. ORCLUS is very similar to the K-means algorithm, except that rather than measuring distances in the full space, distances are measured in subspaces. ORCLUS consists of a number of iterations, where in each successive iteration clusters are refined by peeling off noisy

dimensions for different clusters, and reducing the number of clusters by a series of merge operations. Subspaces are formed in each iteration by performing PCA, that is, computing the covariance matrix for each cluster and selecting a set eigenvectors with the least spread that span the subspace in which the cluster is embedded. A merge operation is carried out when clusters are near each other and have similar directions of eigenvectors. Points are assigned to clusters whose center is closest. The algorithm is computationally intensive $O(d^3)$ due to the computation of the covariance matrix and eigenvectors of each cluster. In addition, ORCLUS suffers from the drawbacks discussed earlier, namely the requirement of the number of clusters and subspace dimensionality as input parameters.

Other top-down methods include FINDIT [55] which is very similar in structure to PROCLUS but based on dimension voting, whereby the algorithm counts the number of dimensions on which points are within a certain threshold distance of each other. It is based on the assumption that the more dimensions are counted the more meaningful the points under consideration are. FINDIT is able to find clusters in subspaces of varying size, and employs sampling to improve performance. COSA (Clustering On Subsets of Attributes) [56] is an iterative method that rather than assigning weights to the dimensions of a cluster assigns them to points. Starting with equally weighted dimensions assigned to each point the algorithm finds the k-nearest neighbors (knn) of each point, and uses them to estimate the weights of each dimension, where higher weights are assigned to dimensions of smaller dispersion. These weights are then used to compute weights for pairs of points, which are in turn used to update the distances used in the knn computation. The process is then repeated until the weights stabilize. The dimensionality of the subspaces in

which clusters are assumed to exist is not specified directly, but the outcome tends to produce clusters of similar dimensionality.

DOC (Density-based Optimal projective Clustering) [57] is another subspace clustering algorithm that cannot really be classified into any of the above two classes of algorithms. It is an iterative algorithm but not a partitioning one as the top-down methods, it uses a grid based approach used by the bottom-up methods but is not incremental as they are and does not rely on the downward closure property of density. DOC is a monte-carlo based algorithm. It uses random sampling to select a small set points, called a discriminating set to determine which dimensions are relevant to a cluster. The sampling is repeated several times and the best results are reported. Like other algorithms, DOC is restricted to axis-parallel subspace clusters, and is heavily dependant on its input parameters. Its running time is linearly proportional to the number of points, but exponential in the number of dimensions.

5.5 A comparison

Subspace clustering algorithms combine clustering techniques and feature evaluation schemes in order to detect clusters embedded in different subspaces of the data. The bottom-up approaches start with clusters embedded in lower dimensionality subspaces and extend them to higher dimensional subspaces by employing an incremental scheme that exploits the downward closure property of density to prune "unworthy" dimensions from the search space. They scale well with the size of the data and its dimensionality, but not with the dimensionality of the subspaces in which the clusters are embedded. This makes them more appropriate for data sets whose clusters are

believed to exist in lower dimensional subspaces. In addition they are typically grid-based, and because of that are extremely sensitive to the cell width or the gird unit size input parameter. Some bottom-up methods alleviate this problem by employing an adaptive gird approach. They allow the concept of overlapping clusters, and are able to identify clusters of various shapes and sizes. The top-down methods on the other hand, use iterative schemes that construct clusters in higher dimensional subspaces, evaluate them, and then peal off the "unworthy" dimensions until some stoping criteria is met, which is typically the dimensionality of the final subspaces. Evaluating the subspaces is typically done using expensive methods such as PCA, which makes this class of algorithms not as efficient as the bottom-up methods. Sampling techniques are usually employed by this class of algorithms to improve performance. But this in turn may degrade cluster quality when the samples or their sizes for example are not chosen correctly. Combined together these "symptoms" make the top-down approaches more appropriate for data sets whose clusters are believed to be larger and embedded in relatively larger subspaces. Due to their nature they do not support the concept of overlapping clusters, and are typically confined to clusters that are spherical in shape. Many of the top-down methods also require the expected number of clusters and their corresponding subspace dimensionality as input. This causes the generated clusters of be of similar subspace dimensionality which is not very realistic. Nonetheless, unlike the bottom-up methods they are not all confined to axis-parallel subspaces, and do allow for the subspaces in which clusters may be embedded to be arbitrarily oriented, which is more realistic.

Chapter 6

Pattern and Correlation Clustering

6.1 Introduction

In recent increasingly important applications of clustering such as gene expression microarray analysis, collaborative filtering, and web mining, object similarity is no longer measured by physical distance, but rather by the behavior patterns objects manifest or the magnitude of correlations they induce under a subset of features. Full dimensional clustering methods including subspace clustering methods focus on grouping objects with similar values, and therefore are inadequate to capture coherent behavior patterns or correlations. A set of points may be located far away from each other yet manifest coherent behavior patterns or induce large correlations among some subset of dimensions. The detection of correlations is a an important data mining task because correlations may reveal a dependency or some cause and effect relationship between the features under consideration. In recent studies these correlations were often discussed and presented in terms of the behavior patterns objects manifest, hence the name *pattern clustering* often associated with methods amid at this type of problem. In gene expression microarray clustering the goal is to identify groups of genes that exhibit similar expression patterns under some subset of

conditions (dimensions), independent of their magnitude, from which gene function or regulatory mechanisms may be inferred. In recommendation or collaborative filtering systems, sets of customers with similar interest patterns need to be identified so that customers' future interests can be predicted and proper recommendations be made. Initial solutions to this problem included either constructing a new data set by various transformations or using the Pearson correlation similarity measure. The transformations either increased the size of the data or masked certain aspects of it. While effective in full-space clustering problems the Pearson correlation measure of similarity is not well suited to discover coherent behavior patterns that exist only in subspaces of the data. In the following we discuss pattern based clustering methods focusing on the application of gene expression microarray analysis. The reason we choose to focus on this application is due to its increasing popularity and importance, and because most of the pattern based clustering methods are aimed at this application.

6.2 Gene Expression Microarray Analysis

Genes are DNA (Deoxyribonucleic acid) sequences that contain the code (instructions) to generate proteins, which are the building blocks of all living cells and that carry out vital organism functions. Gene expression is the process by which cells convert genetic information (DNA sequences) to mRNA (messenger ribonucleic acid) and from mRNA to proteins, as suggested by the *central dogma of molecular biology*. The conversion of DNA encoded information to its mRNA equivalent is called *transcription*. The conversion of mRNA to proteins is called *translation*. The rate of transcription by the cell for different genes varies, and therefore the amount of certain

mRNAs in the cell is a measure of the production speed of a corresponding protein in the cell and the state of the cell. Depending on the environment of the cell (and other factors), different amounts of some proteins are required, hence different concentrations of mRNAs for different genes exist in the cell. The relationship between the amount of mRNA observed under experimental conditions, versus the amount observed under control conditions is called the expression level.

DNA microarray technology allows to globally and simultaneously observe (measure) the expression levels (behavior patterns) of thousands of genes within a number of different experimental samples (conditions). The samples may correspond to different time points or different environmental conditions. In other cases, the samples may have come from different organs, from cancerous or healthy tissues, or even from different individuals. The fundamental assumption in microarray experiments is that the transcriptional events observed are directly related to the experimental conditions in which they are observed.

Elucidating the patterns hidden in gene expression data offers tremendous opportunity for an enhanced understanding of functional genomics. However, analyzing the data obtained from a microarray experiment and extracting biologically relevant knowledge from it has remained a challenging problem. A first step towards addressing this challenge is through clustering to reveal natural structures and identify interesting patterns in the underlying data. Specifically, genes with similar expression patterns (*co-expressed* genes) can be clustered together into groups of similar cellular functions, which may further the understanding of the functions of many genes for which information is unavailable. Moreover, co-expressed genes in the same cluster

	condition 1	condition 2	...	condition d
gene 1	a_{11}	a_{12}	...	a_{1d}
gene 2	a_{21}	a_{22}	...	a_{2d}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
gene N	a_{N1}	a_{N2}	...	a_{Nd}

Figure 6.1: Gene expression matrix.

are likely to be involved in the same cellular process, and a strong correlation of expression patterns between those genes indicates *co-regulation* (regulation by the same cellular process), which may further the understanding of regulating mechanisms in cells.

Gene expression data is typically arranged in a data matrix, where each gene corresponds to one row and each condition to one column, Fig. 6.1. Each element of this matrix a_{ij} represents the expression level of gene i under condition j, and is typically represented by a real number, which is usually the logarithm of the relative abundance of the mRNA of the gene under the specific condition.

Gene expression matrices have been traditionally subject to clustering (using full space clustering methods) in two dimensions: the gene dimension, i.e., clustering rows of the matrix, and the conditions dimension, i.e., clustering columns of the matrix. However the results of the application of standard (full space) clustering methods is limited. These limitations are imposed by the existence of a number of experimental conditions where the activity of genes is uncorrelated. In other words, many activation patterns are common to a group of genes only under a subset of experimental conditions. Basic understanding of cellular processes suggests that subsets of genes

are co-expressed only under certain experimental conditions, but behave almost independently under other conditions. A similar argument can be posed when clustering of conditions is performed. Discovering these local expression patterns is key to uncovering biologically relevant knowledge that is not apparent otherwise. Because of this there has been a move from the traditional clustering approaches to approaches that perform simultaneous clustering on the rows and columns of the gene expression matrix to uncover local expression patterns. This simultaneous clustering, usually referred to as *biclustering*, seeks to find sub-matrices (subsets of genes and subsets of conditions), such that the genes within these sub-matrices manifest highly correlated activities for each of the included conditions in the sub-matrices. This in essence is similar to subspace clustering that seeks to identify groups of similar (in values) objects that exist in subsets of dimensions and not the full set of dimensions, except that in the case of microarray clustering similarity is defined differently.

6.3 The Biclustering Concept

Given a gene expression data matrix $A = (X, Y)$ where X is the genes set and Y the conditions set, the goal of biclustering is to identify biclusters (sub-matrices) $B_k = (I_k, J_k)$, where $I_k \subseteq X$ and $J_k \subseteq Y$, such that each bicluster B_k satisfies some specific homogeneity criteria. The exact characteristics of the homogeneity criteria that a bicluster must satisfy varies from approach to approach and determines the type of biclusters that will be identified by a biclustering algorithm. The more common homogeneity criteria that are used or bicluster models that are assumed by biclustering methods are [58]:

1. Biclusters with constant values.

2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0
2.0	2.0	2.0	2.0	2.0

Figure 6.2: Constant values bicluster.

2. Biclusters with constant values on rows or columns.

3. Biclusters with coherent values.

6.4 Biclusters with Constant Values

The simplest biclustering algorithms identify subsets of rows and subsets of columns in the expression matrix with constant values, Fig. 6.2. This type of clustering is essentially a special case of subspace clustering which seeks to identify objects with similar values (almost constant) in a subset of dimensions. Similar to the model presented in eq. (3.6.1), a bicluster of constant values can be modeled by

$$a_{ij} = \mu + \epsilon, \tag{6.4.1}$$

where a_{ij} is the value of the i-th gene under the j-th condition, μ the profile (constant value) of the bicluster, and ϵ a noise or error term added to reflect the fact that in real data sets the constant value biclusters are rarely perfect. The criteria used to qualify constant value biclusters is typically the variance of the values a_{ij} within the bicluster [59, 60].

6.5 Biclusters with Constant Values on Rows or Columns

A slightly more elaborate biclustering approach seeks to identify subsets of rows and subsets of columns in the expression matrix with constant values on the rows or columns of the data matrix, Fig. 6.3. From a biological perspective these type of biclusters assume that a single cellular process is the main contributor to the expression levels and that the genes or conditions should have constant expression values, but may differ from gene to gene or condition to condition. From a model based view these type of biclusters can be modeled by

$$a_{ij} = \mu + \alpha_i + \epsilon, \tag{6.5.1}$$

and

$$a_{ij} = \mu \times \alpha_i + \epsilon, \tag{6.5.2}$$

where μ is the profile or mean of the bicluster, α_i is the adjustment for the i-th row (gene), which can either be additive (6.5.1) or multiplicative (6.5.2), and ϵ is a noise or error term. Similar models can be described for biclusters with constant values across the conditions. From a correlation point of view these type of biclusters will induce large correlations between the conditions or the genes respectively of the biclusters' expression sub-matrix. Identifying these type of biclusters cannot be done by examining the variance of the values in the bicluster as in the case of constant value biclusters, or by examining the similarities between the rows and columns of the data matrix. One approach to identify these type biclusters is to normalize the the rows or columns of the expression matrix by the row or column means respectively [61]. By that, these biclusters are essentially transformed to the constant value biclusters,

1.0	1.0	1.0	1.0	1.0
2.0	2.0	2.0	2.0	2.0
3.0	3.0	3.0	3.0	3.0
4.0	4.0	4.0	4.0	4.0
5.0	5.0	5.0	5.0	5.0

(a) constant rows

1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0
1.0	2.0	3.0	4.0	5.0

(b) constant columns

Figure 6.3: Biclusters with constant values on rows or columns.

and methods to identify the constant value biclusters can be used.

6.6 Biclusters with Coherent Values

The most sophisticated yet most popular approaches to biclustering seek to identify biclusters with coherent expression behavior on both the rows and columns of their corresponding expression sub-matrices. The most widely studied behavior or expression patterns are the *shift* and *scaling* patterns, which induce only positive correlations. In the collaborative filtering literature clusters manifesting shift patterns are also called *slope one* clusters [62] because of their geometrical orientation (further discussed in chapter 10). In the case of a shift pattern the behavior pattern of one gene under a subset of conditions is offset from another by some constant, whereas in the case of scaling the behavior pattern of one gene is a scaler multiple of another. The scaling pattern is often reduced to the shift by various transformations such as the log transform so that methods aiming at shift biclusters can also be used to identify scaling biclusters. The bicluster matrices depicted in Fig. 6.4 are examples of such biclusters, where each row has the same expression pattern as the other except that it is either translated (shifted) from the other by a constant or is a scalar multiple of

1.0	2.0	5.0	0.0	3.0
2.0	3.0	6.0	1.0	4.0
4.0	5.0	8.0	3.0	6.0
7.0	8.0	11.0	6.0	9.0
8.5	9.5	12.5	7.5	10.5

(a) shift bicluster

1.0	2.0	5.0	0.5	3.0
2.0	4.0	10.0	1.0	6.0
3.0	6.0	15.0	1.5	9.0
4.0	8.0	20.0	2.0	12.0
0.5	1.0	2.5	0.25	1.5

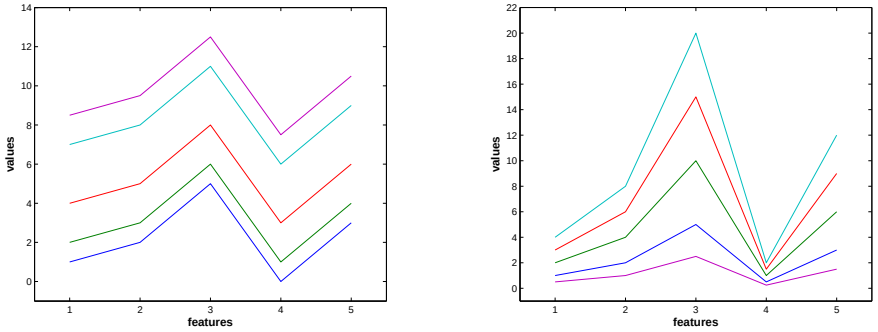
(b) scale bicluster

Figure 6.4: Biclusters with coherent expression behavior.

the other.

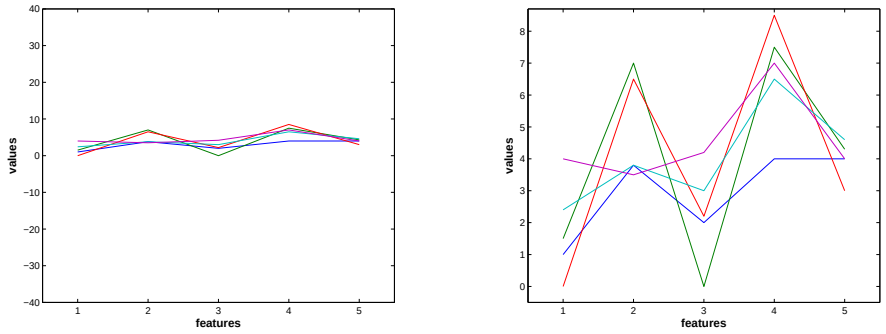
A *parallel coordinate* plot (2-dimensional plot) is a visualization technique that is typically used to demonstrate and emphasize symmetry or cohesion in behavior patterns of objects under consideration. Consequently, these type of plots have been extensively used in gene expression analysis to investigate and demonstrate co-expressed genes. In the parallel coordinate plot each polygonal line corresponds to an object (a gene), the x-axis to the features of the object (conditions), and the y-axis to the value of each object (expression level) under a specific feature (condition). Fig. 6.5 shows two such plots corresponding to the expression matrices depicted in Fig. 6.4, illustrating the shift and scaling behavior patterns respectively of the five objects under consideration. Notice the symmetry in the behavior patterns of the five objects. Fig. 6.6 in contrast is a parallel coordinate plot of a set of objects that have similar values, i.e., are physically located close to each other. In the context of clustering this set of objects will be considered to form a "classical" cluster or a so called constant columns bicluster. Although they seem to be manifesting symmetric behavior patterns (Fig. 6.6(a)), a closer look (Fig. 6.6(b)) shows they are not, and will therefore not be considered a cluster with coherent behavior patterns.

As mentioned before the problem of clustering objects or genes by their behavior



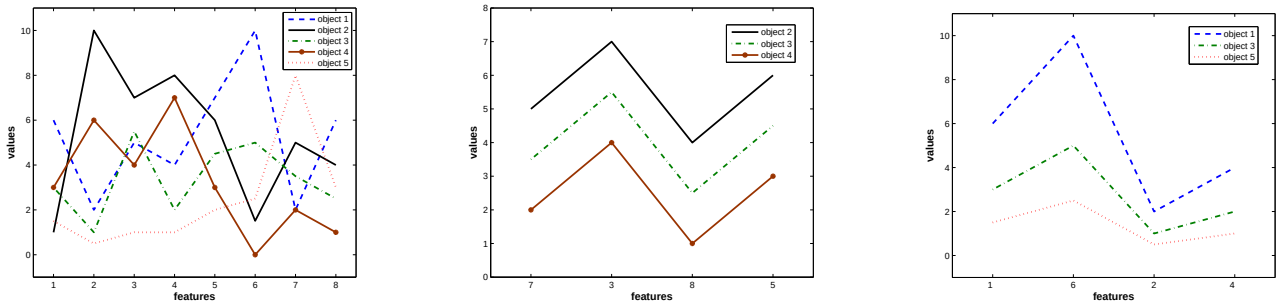
(a) shift pattern (b) scaling pattern

Figure 6.5: Parallel coordinate plots of the shift and scaling biclusters depicted in Fig. 6.4.



(a) farther look (b) closer look

Figure 6.6: Parallel coordinate plots of a regular cluster (objects with similar values).



(a) no patterns in the full space (b) shift pattern in a subspace (c) scaling pattern in a subspace

Figure 6.7: Patterns evident only in subspaces.

patterns is exacerbated when the coherent patterns are evident only in a smaller subset of features or conditions. Fig. 6.7 illustrates this problem motivating the need for biclustering methods. The figure shows a data set with 5 objects and 8 features where no patterns among the 5 objects are visibly explicit in the full set of features (Fig. 6.7(a)). However, if we pick a subset of these features, $\{7, 3, 8, 5\}$, a shift pattern induced by 3 objects is revealed (Fig. 6.7(b)), which may have not been considered if the patterns were sought in the full set of features. Similarly, a scaling pattern is revealed when focusing only on features $\{1, 6, 2, 4\}$ (Fig. 6.7(c)). Also notice that one object (number 3) is involved in two different pattern clusters, in the context of gene expression analysis this may indicate that the gene belongs to two different functional classes or two different cellular processes. In most cases however different objects (possibly some in common) will exhibit coherent behavior in a different subset of features.

6.6.1 The Additive Model

The majority of biclustering algorithms aiming at shift patterns assume an underlying *additive model* to describe a bicluster. The origin of this model comes from a theory known as the *analysis of variance* (ANOVA), which attempts to decompose the total variation in the data into contributions from various sources called factors [63]. In particular, the additive model used in biclustering is part of the *two-factor analysis of variance* model. ANOVA models, just like regression models, assume a response variable Y, in our case a (the expression level), that is influenced by other predictor variables called factors that generally take on finitely many values called levels. The general goal in ANOVA studies is to estimate and test the effects of the factor levels on the mean of the response variable, and find possible interactions between the different factors. In the two-factor additive model the sources of variation of the response variable are assumed to come from two factors, e.g., genes and conditions. Using the same notation used in eq. (6.4.1)-(6.5.2), let α and β denote these two factors, having I and J levels respectively. Let a_{ij} denote the value of the response variable under the i-th level of factor α and the j-th level of factor β, and let μ be the overall mean of a. In our case, α represents the genes factor, β the conditions factor, I the number of genes, J the number of conditions, and a_{ij} the expression level of i-th gene under the j-th condition. The basic ANOVA model assumes that the value of the response variable a is the sum of a deterministic (non-random) term, and a stochastic (random) error term that is typically normally distributed with mean 0 and variance σ^2. In the basic two-factor model the deterministic term is broken into the overall mean plus the residual effect due to the i-th level of factor α (i-th gene) denoted by α_i and the j-th level of factor β (j-th condition) denoted by β_j. Hence,

according to the additive model a coherent shift bicluster is modeled by

$$a_{ij} = \mu + \alpha_i + \beta_j + \epsilon. \tag{6.6.1}$$

The basic two-factor model assumes an additive effect of the factors, which is often called the *assumption of additivity*. The meaning of this assumption in the context of gene expressions is as follows. Suppose a particular gene under a particular condition has an expression level which is c units more than some other gene under the same condition. Then by the assumption of additivity, it must also be true that for each of the other $J - 1$ conditions the first gene has an expression level that is c units larger than the other gene. Likewise, suppose that a particular gene under a certain condition has an expression level that is c units larger than under a different condition. Then by the same assumption all other $I - 1$ genes have an expression level that is c units larger under the first condition than the other condition. This explains why models that are based on the assumption of additivity are in essence modeling a shift behavior pattern. When the factor levels are sampled from a population of levels, e.g., the set of genes selected for the experiment are only a subset of the total number of genes, the factors are assumed to make random contributions. That is, α and β are now considered random variables.

ANOVA models are usually evaluated by a measure representing the sum of squared deviations of the response variable from its mean denoted by SST. Focusing on the two-factor additive model, let

$$\bar{a}_i = \frac{1}{|J|}\sum_{j=1}^{J} a_{ij}, \quad \bar{a}_j = \frac{1}{|I|}\sum_{i=1}^{I} a_{ij}, \quad \bar{a}_{IJ} = \frac{1}{|I||J|}\sum_{i=1}^{I}\sum_{j=1}^{J} a_{ij}, \tag{6.6.2}$$

be the sample mean of the response variable a under the i-th level of factor α, the sample mean of the response a variable under the j-th level of factor β, and the

sample mean of a respectively. Then the sum of squared deviation of the response variable from its mean is given by

$$\begin{aligned} SST &= \sum_{i=1}^{I}\sum_{j=1}^{J}(a_{ij}-\bar{a}_{IJ})^2 \\ &= \sum_{i=1}^{I}\sum_{j=1}^{J}[(\bar{a}_i-\bar{a}_{IJ})+(\bar{a}_j-\bar{a}_{IJ})+(a_{ij}-\bar{a}_i-\bar{a}_j+\bar{a}_{IJ})]^2 \\ &= I\sum_{j=1}^{J}(\bar{a}_i-\bar{a}_{IJ})^2+J\sum_{i=1}^{I}(\bar{a}_j-\bar{a}_{IJ})^2+\sum_{i=1}^{I}\sum_{j=1}^{J}(a_{ij}-\bar{a}_i-\bar{a}_j+\bar{a}_{IJ})^2. \end{aligned} \tag{6.6.3}$$

Hence, SST can be partitioned into three smaller sum of squares denoted SSF_α, SSF_β, and SSE respectively that model the sum of squared deviations in terms of deviations arising from three different sources. The deviations that are due to the two factors α and β, and the deviation due to the error term. Since the *maximum likelihood estimates* (MLE's) of μ, α_i, and β_j are $\bar{a}_{IJ}$, $\bar{a}_i-\bar{a}_{IJ}$, and $\bar{a}_i-\bar{a}_{IJ}$ respectively, the MLE of $E[a_{ij}]=\mu+\alpha_i+\beta_j=\bar{a}_i+\bar{a}_j-\bar{a}_{IJ}$. Therefore SSE can be rewritten as

$$SSE=\sum_{i=1}^{I}\sum_{j=1}^{J}(a_{ij}-\bar{a}_i-\bar{a}_j+\bar{a}_{IJ})^2=\sum_{i=1}^{I}\sum_{j=1}^{J}(a_{ij}-E[a_{ij}])^2, \tag{6.6.4}$$

which is the variance of the response variable a. The majority of biclustering algorithms use SSE as defined in eq. (6.6.4), to measure the coherence in expression behavior of a set of genes under a set of conditions and to qualify biclusters following a shift pattern, where $SSE=0$ indicates that a perfect bicluster has been identified. In the biclustering literature SSE is typically called the *mean squared residue score* and denoted by $H(I,J)$ or MSRS. It should be strongly emphasized that the mean squared residue score can only be used if the biclusters are assumed to follow the additive model or equivalently the shift pattern which is typically not the case.

6.6.2 The Multiplicative Model

Some biclustering approaches assume that the biclusters follow a multiplicative model or equivalently follow the scaling pattern. According to the multiplicative model the expression level of the i-th gene under the j-th condition a_{ij} can be modeled by

$$a_{ij} = \mu \times \alpha_i \times \beta_j \times \epsilon, \tag{6.6.5}$$

where μ, α_i, β_j, and ϵ are defined in same way as in the additive model, but now the factors effect the typical value (mean) of the bicluster in a multiplicative manner (the error term ϵ can also be modeled in an additive manner). Many biclustering approaches agree that the multiplicative model is a more appropriate model than the additive model to describe the expression levels of genes as it allows for more abrupt changes in the expression level, and because genes tend to express themselves on different scales. Nonetheless, by using the log transform the multiplicative model can be reduced to the additive model. As a consequence, many biclustering approaches designed to identify biclusters which are based on the additive model propose to first preprocess the data by computing the log of expression levels in order to identify biclusters following the multiplicative model or the scaling pattern.

6.7 Correlation

In probability and statistics, *correlation* is a measure that is used to quantify the strength and direction of a relationship, typically linear, between two random variables, or the departure of two variables from independence. Although correlation does not imply causation (causation implies correlation), it is generally used to hypothesize some sort of cause and effect between two variables. There are several measures

of correlation called *correlation coefficients*, each applicable to a different situation (types of variables) and each with its strengths and weaknesses. The most widely studied and used correlation coefficient is the *Pearson product-moment correlation coefficient* [64] intended to capture linear relationships. Given two random variables X and Y the Pearson product-moment correlation coefficient ρ_{XY} is defined as

$$\rho_{XY} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathrm{E}[(X - \mathrm{E}[X])(Y - \mathrm{E}[Y])]}{\sqrt{\mathrm{E}[(X - \mathrm{E}[X])^2]}\sqrt{\mathrm{E}[(Y - \mathrm{E}[Y])^2]}}, \tag{6.7.1}$$

and ranges in $[-1, 1]$ where $\rho_{XY} = 1$ indicates perfect positive correlation, $\rho_{XY} = -1$ indicates perfect negative correlation, and $\rho_{XY} = 0$ indicates no correlation. When more than two variables are involved in a correlation analysis, a correlation coefficient ρ_{ij} is computed for each pair of variables (features) i, j, and the results are typically summarized in a correlation coefficient square matrix where the (i, j) entry corresponds to ρ_{ij}.

Proposition 6.7.1. *A perfect shift bicluster (pattern) induces perfect positive correlations among a subset of features.*

Proof: According to eq. (6.6.1) a perfect (no error term) shift pattern is modeled by $a_{ij} = \mu + \alpha_i + \beta_j$. To compute correlation we first need to restate the model in terms of features and random variables. Let a_m denote a random variable corresponding to the m-th feature, dimension, or the m-th level of factor β effecting the values in a shift bicluster. Then value of the m-th feature can be modeled by $a_m = \mu + \alpha + \beta_m$, where μ as before is a constant, β_m is now also a constant (by definition the effect of factor β remains constant under the m-th level), and where α is a random variable denoting the α factor, i.e., the values of the bicluster under the different levels of factor α and the m-th level of factor β. To simplify the model let $\mu_m = \mu + \beta_m$,

yielding

$$a_m = \mu_m + \alpha.$$

Similarly the values taken by the n-th feature can be modeled by

$$a_n = \mu_n + \alpha.$$

Having two (arbitrary) features the correlation between them can now be computed. Becuase

$$\mathrm{E}[a_m] = \mu_m + \mathrm{E}[\alpha], \qquad \mathrm{E}[a_n] = \mu_n + \mathrm{E}[\alpha],$$

$$\mathrm{Var}[a_m] = \mathrm{Var}[a_n] = \mathrm{Var}[\alpha],$$

and

$$\begin{aligned} \mathrm{Cov}(a_m, a_n) &= \mathrm{E}[(a_m - \mathrm{E}[a_m])(a_n - \mathrm{E}[a_n])] \\ &= \mathrm{E}[(\alpha - \mathrm{E}[\alpha])(\alpha - \mathrm{E}[\alpha])] \\ &= \mathrm{E}[(\alpha - \mathrm{E}[\alpha])^2] \\ &= \mathrm{Var}[\alpha], \end{aligned}$$

the correlation ρ_{mn} (as defined by eq. 6.7.1) between any pair of features in a bicluster is equal

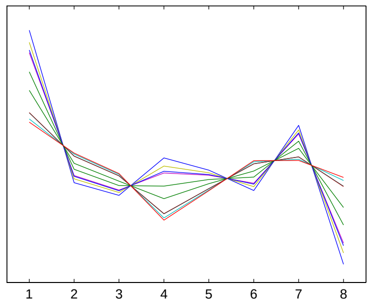
$$\rho_{mn} = \frac{\mathrm{Cov}(a_m, a_n)}{\sqrt{\mathrm{Var}[a_m]}\sqrt{\mathrm{Var}[a_n]}} = \frac{\mathrm{Var}[\alpha]}{\mathrm{Var}[\alpha]} = 1,$$

a perfect positive correlation. □

Along the same lines a similar proof can be used to show that a shift pattern induces perfect correlations between the objects of a bicluster, i.e., the features in this case are considered to be the objects. In addition a similar proof can be used to prove the same results for scaling pattern clusters. However, in this case the

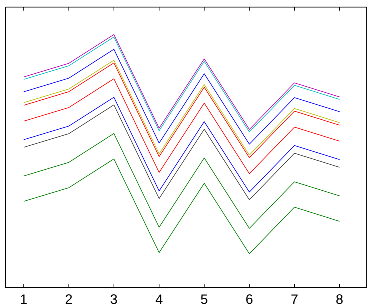
correlations are positive only because the terms involved in the multiplicative model (eq. 6.6.5) are assumed to be positive, a reasonable assumption in the case of gene expressions since mRNA levels, so as scaling factors cannot be negative.

Therefore, from a correlation point of view the shift and scaling patterns induce only positive correlations. However large negative correlations are just as important and information carrying as large positive correlations. Most biclustering methods completely overlook the possibility that negative correlations may be present in the data [65, 66]. Negative correlations express the fact that when one gene has a higher expression level another gene will have a lower expression level, and that negative regulatory effects exist as well as positive regulatory effects. Fig. 6.8 shows a comparison of the correlations induced on a set of features by four types of pattern behaviors, where each pattern is presented by an axis-parallel plot and the corresponding correlation coefficient matrix induced by the objects manifesting the pattern. Notice that the pattern inducing negative correlations (Fig. 6.8(a)) does not show the same type of pattern symmetry as the shift and scaling patterns (Figs. 6.8(a)(b)). Nonetheless its correlation coefficient matrix reveals perfect correlations which are both positive and negative as apposed to the shift and scaling patterns which are associated with perfect positive correlation coefficient matrices. To contrast the three correlation inducing patterns Fig. 6.8(d) shows the correlation induced by a set of objects that have similar values (have small distances between each other). Although they seem to be manifesting symmetric behavior patterns (which they are not when looked at closely), they do not induce any large correlations, and therefore are not considered to exhibit coherent behavior.



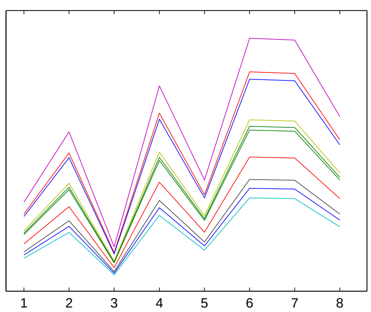
+1.00	-1.00	-1.00	+1.00	+1.00	-1.00	+1.00	-1.00
-1.00	+1.00	+1.00	-1.00	-1.00	+1.00	-1.00	+1.00
-1.00	+1.00	+1.00	-1.00	-1.00	+1.00	-1.00	+1.00
+1.00	-1.00	-1.00	+1.00	+1.00	-1.00	+1.00	-1.00
+1.00	-1.00	-1.00	+1.00	+1.00	-1.00	+1.00	-1.00
-1.00	+1.00	+1.00	-1.00	-1.00	+1.00	-1.00	+1.00
+1.00	-1.00	-1.00	+1.00	+1.00	-1.00	+1.00	-1.00
-1.00	+1.00	+1.00	-1.00	-1.00	+1.00	-1.00	+1.00

(a) negative correlation inducing pattern



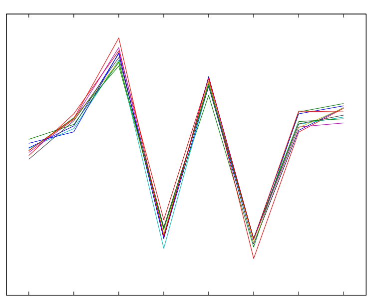
+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00
+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00
+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00
+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00
+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00
+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00
+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00
+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00

(b) shift pattern



+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00
+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00
+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00
+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00
+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00
+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00
+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00
+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00	+1.00

(c) scaling pattern



+1.00	-0.63	-0.19	-0.39	+0.11	+0.34	+0.29	-0.01
-0.63	+1.00	+0.16	+0.73	-0.10	-0.62	-0.18	+0.07
-0.19	+0.16	+1.00	-0.08	+0.67	+0.01	+0.09	-0.17
-0.39	+0.73	-0.08	+1.00	-0.09	-0.64	-0.16	+0.20
+0.11	-0.10	+0.67	-0.09	+1.00	+0.00	-0.10	-0.14
+0.34	-0.62	+0.01	-0.64	+0.00	+1.00	+0.37	-0.49
+0.29	-0.18	+0.09	-0.16	-0.10	+0.37	+1.00	+0.25
-0.01	+0.07	-0.17	+0.20	-0.14	-0.49	+0.25	+1.00

(d) similar value pattern

Figure 6.8: A comparison of four behavior patterns and the correlations they induce among the eight features of the data.

6.8 Biclustering Algorithms

Cheng et al. [67] were the first to use the bicluster term and introduced the concept of coherent biclusters in the context of gene expression analysis. They define a bicluster as a sub-matrix having mean squared residue score $H(I, J)$ less than some threshold δ, and aimed at finding the largest such biclusters. To find a bicluster, the mean squared residue score $H(I, J)$ is computed for each row/column addition or deletion, and the action that decreases $H(I, J)$ the most was applied. The process is repeated until $H(I, J)$ stops decreasing or until $H(I, J) \leq \delta$, in which case a bicluster is returned. After a bicluster is identified it is removed by replacing its elements in the corresponding data matrix with random values. This process then continues until a pre-specified number of biclusters have been identified. The algorithm has time complexity $O((n + m)mn)$, where n and m are the number of genes and conditions respectively, which may be impractical for large expression matrices. An improvement is proposed by the same authors which is based on multiple row/column additions/deletions, and which has complexity $O(mn)$. The algorithm has several drawbacks. The first is that the number of clusters must be specified in advance. Second, the removal of identified clusters prevents it from identifying overlapping biclusters. Third, it assumes that the biclusters follow the additive model (eq. (6.6.1)), and therefore is limited to finding only these type of clusters.

Yang et al. [68] proposed an improvement to the biclustering algorithm and presented the move-based algorithm FLOC (FLexible Overlapping biClustering). They define a δ-cluster that is able to cope with missing values and avoid the random fillings used by the biclustering algorithm. They use an occupancy threshold α to limit the amount of missing values in a δ-cluster, and use the mean squared residue score,

computed only on existing (not missing) values, to qualify a δ-cluster. The FLOC algorithm also requires a number of clusters to be pre-specified, and starts with k randomly selected sub-matrices (clusters). It then iteratively adds/removes rows or columns in and out of the cluster to lower the score, until a minimum is reached. The complexity of FLOC is $O((n+m)nmkl)$ where l is the number of iterations.

Weng et al. [69] introduced the *pCluster* model which also aims at identifying shift pattern biclusters but does not use the mean squared residue score. They consider a 2×2 sub-matrix of two objects i, j and two attributes m, n and define the *pScore* as a criteria for identifying shift patterns, where

$$pScore = |(a_{im} - a_{in}) - (a_{jm} - a_{jn})|.$$

$pScore \leq \delta$ means that the change of values on two attributes between two objects is confined by a pre-defined threshold δ. And if such a confinement applies to every pair of objects and attributes in a potential cluster then a shift cluster has been detected. The pCluster algorithm uses a depth-first search approach facilitated by an MDS (maximum dimension set) principle to perform pair-wise clustering. The main advantages of the algorithm are that unlike other algorithms it is deterministic, and therefore does not miss any qualified clusters. It is able to discover overlapping clusters, and is resilient to outliers.

Maple (Maximal Pattern-Based Clustering) [70] is an extension to the pClsuter algorithm that is more efficient by skipping trivial subclusters and pruning non-promising cluster candidates earlier.

Other biclustering algorithms include, CTWC (Coupled Two-Way clustering) [61] which repeatedly performs one way (regular) clustering on the rows and columns of

the data, each used to cluster the other, using any clustering algorithm and similarity measure. Lazzeroni et al. [71] introduced the *plaid model* which regards gene expression data as a sum of multiple "layers" (clusters), where each layer represents a particular biological process with only a subset of genes and conditions involved. Under this model it is assumed that if a gene participates in several processes, then its expression level is the sum of terms involved in the individual processes. The clustering process is performed using the EM algorithm to estimate the parameters of the model. An approach based on projected clusters called HARP was presented in [72], and uses a measure called "relevance index" based local and global variance to measure the quality of clusters.

6.9 Problems with current State of the Art Approaches

In recent studies [65, 66, 73, 74] it has been suggested that other types of information carrying patterns such as patterns inducing negative correlations are completely overlooked by most clustering methods, and that current state of the art algorithms are not flexible enough to mine different types of patterns such as the shift and scaling simultaneously, as it is more likely that different types of behavior patterns co-exit in the data. While there is no consensus on what types of patterns should be considered meaningful, in practice pattern based clustering algorithms postulate a unique underlying "globally expressed" pattern or model, while overlooking or rejecting the possibility that other types of information carrying patterns may exit in the data. This in turn typically leads to a large bias in the results, and thus more general

methods would be very beneficial. From our point of view, since coherence of patterns or equivalently correlation inducing patterns are the prime objective, methods aiming at identifying clusters that induce large correlations among subsets of features should be devised to address this challenge.

Another problem is that many clustering approaches rely on one or more data transformations to the data as preprocessing steps in order to highlight certain aspects of the data, or to support underlying assumptions about the characteristics (model) of the clusters, and their underlying distribution. These steps can dramatically effect study conclusions when the assumptions do not hold or when different portions of the data support different assumptions. Among the more popular transformations are the log transform which is used to convert scaling changes into additive increments, row normalization, which standardizes each row with mean zero and variance one, and the subtraction of row means used to eliminate the additive effect and facilitate methods that rely on euclidian distance to measure similarity among objects. It is well accepted that these methods are not always helpful since different patterns typically localize to different clusters, and any one assumption about the nature of the pattern, such as whether it contains a bias or a scaling effect does not hold true across all objects or attributes. Moreover, some of the transformations such as row normalization and mean subtraction will be heavily effected by outliers, noise, or irrelevant dimensions of different clusters. Figure 6.9 demonstrates the problem. The figure shows four different patterns which may co-exist in data undergoing two types of transformations. Assuming a shift pattern (top row), then a log transformation will convert it into a scaling pattern, while a row mean subtraction will as expected convert it into a zero bias pattern, which is a similar values or regular cluster. Hence, an algorithm that

assumes an additive effect will likely overlook this cluster, while an algorithm that searches for regular clusters has the potential to detect it. On the other hand if the original form of a cluster is the scaling pattern (second row), then the log transform as expected will convert it to an additive pattern, but a row mean substraction will not eliminate the scaling pattern. Therefore an algorithm searching for canonical clusters will not be able to identify it since the physical distance between the objects may remain large after the transformation, while an algorithm that assumes the additive model will. Assuming the pattern is a negative correlation inducing pattern (third row), then none of the transformations will be able to convert it into a pattern which can be identified by existing methods. Lastly, if the original form of the cluster is of a regular cluster (bottom row) then a log transform may amplify the difference between objects on a subset of attributes, which may lower the number of attributes originally associated with the cluster and transform it to a subspace cluster, while a row mean subtraction will not change the clusters' shape. Although not shown, we note that an application of the transformation that will standardize each row to mean zero and variance one will as expected transform the first, second, and fourth patterns into a similar values pattern or regular cluster which can be identified with full-space or subspace clustering methods. However, the negative inducing pattern will not be effected by this transformation.

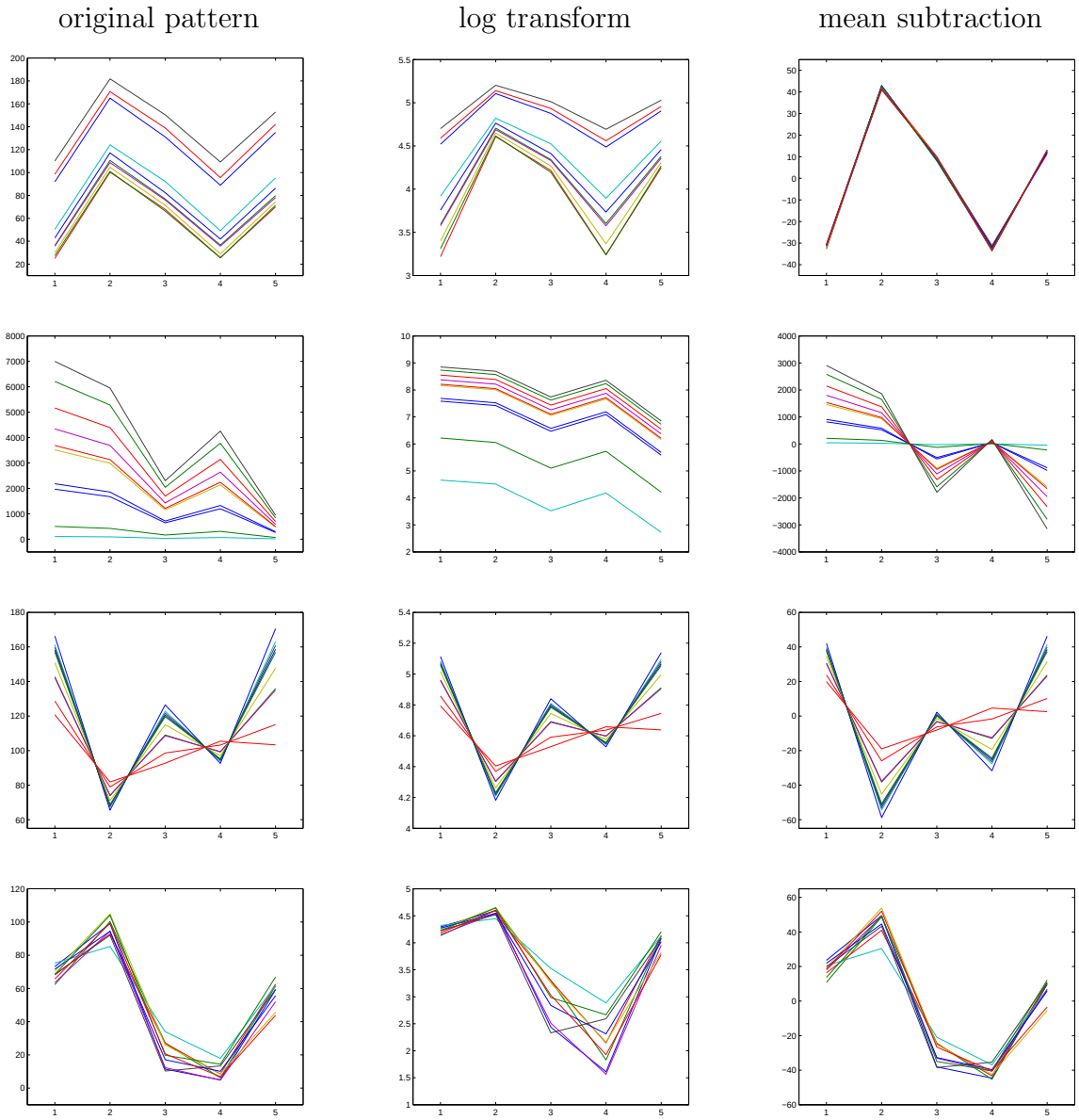


Figure 6.9: Applying transformations to four different patterns.

Chapter 7

The Linear Manifold Cluster Model

7.1 The Model

As mentioned in chapter 2 a linear manifold is a translated subspace, which geometrically can be visualized as a line, plane, hyperplane, etc. In the following we present a formal stochastic *linear manifold cluster model* that describes a "process" that generates sets of points (clusters) that are embedded in or fit bounded linear manifolds. Process models typically consist of two parts, a deterministic and a stochastic. The deterministic part describes the "core" description of the process, the "signal", or the "mathematical structure" that a population of points are assumed to fit. In our case it will describe the linear manifold which the cluster points fit. In reality it is rare that points perfectly fit the deterministic part of a model. There are several reasons for this phenomena which the stochastic part of a model attempts to explain. The most prominent are errors due to the measurement process, noise, or the fact that the reality is too complex to be exactly explained by a model, and can only be approximated. Besides specifying the degree to which the population of points deviate from the deterministic part of a model, in some cases the stochastic part also

describes the distribution of the population within the structure that is modeled by the deterministic part. In either case a probability distribution is associated with the stochastic part. In our case the stochastic part will describe both the degree to which the a set of points deviates from the linear manifold, i.e., the distribution of points away from the manifold, and the distribution of points on the manifold.

Relevant to clustering, the linear manifold cluster model which we propose has the following properties:

- The points in each cluster are embedded in a lower dimensional linear manifold of finite extent.
- The intrinsic dimensionality of the cluster is the dimensionality of the linear manifold, and is generally much smaller than the dimensionality of the data.
- The manifold is arbitrarily oriented.
- In the orthogonal complement space to the manifold the points form a compact densely populated region.
- The points in the cluster induce correlations or linear dependencies between the attributes of the data.

Let X be a set of d-dimensional points that form a cluster, $\mathbf{x}$ be a $d \times 1$ vector representing some point in X, $\mathbf{b}_1, \ldots, \mathbf{b}_d$ be a set of orthonormal vectors that span a d-dimensional space, B be a $d \times k$ matrix whose k columns are a subset of the vectors $\mathbf{b}_1, \ldots, \mathbf{b}_d$, and $\overline{B}$ be a $d \times d - k$ matrix whose columns are the remaining vectors.

Definition 7.1.1 (Linear Manifold Cluster Model)**.** *Let* $\boldsymbol{\mu}$ *be some point in* $\mathbb{R}^d$*,* $\boldsymbol{\lambda}$ *be a zero mean* $k \times 1$ *random vector whose entries are i.i.d.* $U(-R/2, +R/2)$ *where* R *is the range of the data, and* $\boldsymbol{\psi}$ *be a zero mean* $d - k \times 1$ *random vector with small variance independent of* $\boldsymbol{\lambda}$*. Then each point* $\mathbf{x} \in X$*, a linear manifold cluster is modeled by,*

$$\mathbf{x} = \boldsymbol{\mu} + B\boldsymbol{\lambda} + \overline{B}\boldsymbol{\psi}. \tag{7.1.1}$$

Explanation: the idea is that each point in a cluster lays close to a k-dimensional linear manifold of finite extent, which is defined by μ - a translation vector, the subspace spanned by the columns of matrix B, and the range parameter R. Since

$$\mathrm{E}[\mathbf{x}] = \mathrm{E}[\boldsymbol{\mu} + B\boldsymbol{\lambda} + \overline{B}\boldsymbol{\psi}] = \boldsymbol{\mu} + B\mathrm{E}[\boldsymbol{\lambda}] + \overline{B}\mathrm{E}[\boldsymbol{\psi}] = \boldsymbol{\mu} + \mathbf{0} + \mathbf{0} = \boldsymbol{\mu}$$

the cluster mean is $\boldsymbol{\mu}$. On the manifold the points are assumed to be uniformly distributed in each direction (the k column vectors of B) according to $U(-R/2, +R/2)$. However this assumption is not binding, and the uniform distribution can be replaced by any other distribution with symmetric support. The reason we chose the uniform distribution to model the points on the manifold is because typically no prior knowledge is available, and in such cases data is assumed to be "random", i.e., uniformly distributed. It is in this manifold that the cluster is embedded, and therefore the intrinsic dimensionality of the cluster will be k. Since in reality the points will rarely perfectly fit a linear manifold, the third component of eq. (7.1.1) models a small random error associated with each point on the manifold. The idea is that each point may be perturbed in directions that are orthogonal to the subspace spanned by the columns of B, that is the subspace defined by the $d-k$ columns of $\overline{B}$. We model this behavior by requiring that $\boldsymbol{\psi}$ be a $(d-k) \times 1$ random vector, normally distributed according to $N(\mathbf{0}, \Sigma)$, where the largest eigenvalue of Σ is much smaller than R, the range of the data. Otherwise the "signal" cannot be distinguished from the noise.

Hence, the error like in many other typical models is modeled by a Gaussian. In addition, since the variance along each dimension orthogonal to the manifold is now much smaller than the range R of the manifold, the points are likely to form a compact and densely populated region that can be used to cluster the data. The concept of the error term can be visualized by a 1D manifold which transforms from a line into a thin cylinder after the addition of an error term .

Fig. 7.1 is an example of data set modeled by eq. (7.1.1). The data set contains three non-overlapping linear manifold clusters C_1, C_2, C_3 each consisting of 1000 points. C_1, C_2 which are almost planar and parallel to each other are embedded in $2D$ linear manifolds. Their points are uniformly distributed in the manifold and they include a small error term in the space complementary to the manifold. Similarly, C_3 an elongated line-like cluster, is embedded in a $1D$ linear manifold with an error element in the $2D$ space complementary to the manifold. All three manifolds are arbitrarily oriented, and are translated from the origin.

7.2 Motivation

The motivation and associated advantages of the linear manifold cluster model and the linear manifold clustering paradigm may be described from different points of view, some of which are more conceptual and some which are more practical. In the following we provide a description of each of these views.

- Classical clustering algorithms are based on the concept that a cluster center is a single point, and that clusters are sets of points compact around this central point. Most approaches to clustering neglect to consider the possibility that a cluster center may be associated with a more complex structure such as a linear

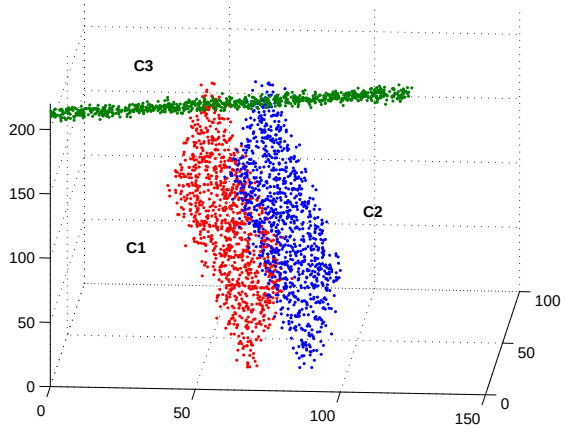


Figure 7.1: Sample data set of three non-overlapping linear manifold clusters, each of which is embedded in a different linear manifold of one ($C3$) or two dimensions ($C1, C2$).

manifold. The linear manifold clustering paradigm introduces the concept of linear manifold clusters which are groups of points compact around a linear manifold.

- The linear manifold cluster model is a generalization of many other more specific cluster models, i.e., other cluster models are instances of the linear manifold cluster model. For example, the classical cluster model assumes that clusters are associated or compact around single points. A zero-dimensional linear manifold is a point. Hence, based on eq. (7.1.1) by making k (the dimensionality of the manifold) equal zero the second component of the model vanishes, and each point in a classical cluster can thus be modeled by $\mathbf{x} = \boldsymbol{\mu} + \overline{B}\boldsymbol{\psi}$ where $\overline{B}$ is simply the identity matrix and $\boldsymbol{\psi}$ a random vector perturbing each point from the center $\boldsymbol{\mu}$.

Subspace clusters can also be modeled using the framework of eq. (7.1.1). Each point in a subspace cluster can be modeled exactly using eq. (7.1.1), taking into account that subspace clustering algorithms focus their clustering effort on the space spanned by the column vectors of $\overline{B}$ in which the points as mentioned before form a compact and densely populated region. When these algorithms are restricted to axis-parallel subspaces, they assume that both B and $\overline{B}$ contain columns of the identity matrix. However, as mentioned in section 5.2 examination of real data often shows that points tend to get aligned along arbitrarily oriented subspaces, and that most subspace clustering algorithms can only identify axis-parallel subspace clusters. The linear manifold model also captures arbitrarily oriented subspace clusters. The only difference between these clusters and the axis-parallel ones in the context of the model is that the matrices B and $\overline{B}$ no longer contain the columns of the identity matrix, but the vectors spanning the arbitrarily oriented subspace in which the clusters are embedded.

Pattern clusters are also instances of linear manifolds. Generally speaking, pattern clusters when confined to the space of relevant dimensions manifest themselves as groups of points compact around a line, which is a one-dimensional linear manifold. In the full space they form linear manifolds of higher dimensionality. Pattern clusters as instances of linear manifolds will be discussed in more detail throughout chapter 10.

This unification of clustering paradigms under one model provides a more general framework in which cluster analysis may be performed. More specifically, a particular cluster structure need not be imposed on the data by a clustering

algorithm any more, in a sense allowing for less assumptions to be made and more freedom for the data to "speak for itself". In the case of pattern clustering or more specifically gene expression clustering this matter was stated more drastically; it has been suggested that other types of information carrying patterns which are also instances of linear manifolds are completely overlooked by most clustering methods, and that current state of the art algorithms are not flexible enough to mine different patterns simultaneously.

- A good clustering scheme is one which provides "scientific content", that is, a probability model which describes the underlying population, and which statistical inference such as predictions can based on. One of the ultimate goals of clustering is not only to reveal structure but to understand the underlying mechanism or "process" which generated the structure. In other words, model the population of points that may have formed the structure discovered by a clustering algorithm. Describing the population by a model based on the sample on which the clustering is applied permits us to gain insights and to learn the most important aspects of the population. Such a model should also provide us the capability of making predictions that are consistent with the data. As stated by physicist Richard Feynman "It is whether or not the theory gives predictions that agree with experiment. It is not a question of whether a theory is philosophically delightful, or easy to understand, or perfectly reasonable from the point of view of common sense" [75]. Lacking scientific content clustering is merely a visualization tool providing different views of the data. Most clustering methods focus only on the grouping and lack the ability to provide any scientific content.

The linear manifold clustering paradigm proposed in this thesis is essentially a model-based clustering paradigm which is based on the model described in section 7.1, and as such "enjoys" the advantages the model-based methods have over other clustering paradigms. Because of its solid mathematical foundation the clustering results are easer to interpret. A mixture of individual linear manifold models can easily be constructed to describe the data, the process generating it, and to provide an estimate of the underlying distribution of the population. Based on this model probabilistic classifiers can be built, and hence predictions for new points can be made. In addition the probabilistic predictions (probability that a point belongs to a certain class or cluster) can be used as a basis for "fuzzy" or "soft" clustering. A more formal description of this idea will be presented in chapter 9-"Data Description, Density Estimation, and Classification".

Because of the probabilistic setting on which the linear manifold clustering paradigm is based, statistical methods such as hypothesis testing, permutations tests, maximum likelihood estimation, etc., can be applied to cluster validity problems making them more tractable.

- The detection of correlations between different features of the data is a very important data mining task. Correlations can be used to reduce the dimensionality of the data by eliminating redundant (correlated) features. Large correlations correspond to approximate linear dependencies between two of more features. Hence, correlations can be used to learn linear dependencies from which causalities may later be hypothesized. Yet another use of correlations, is the identification coherent behavior patterns among a set of objects. Large correlations

imply that coherent behavior patterns between objects exit in the data. These patterns can be used to learn coherent behavior classes such as in the case of DNA microarray analysis, which can later be used to make predictions about future behavior such as in the case of recommendation systems. Correlations or linear dependencies can take on many forms and can be arbitrarily complex, one or more features may depend on a combination of several other features. However, common to all these forms of correlation and linear dependencies, is that in the data space they manifest themselves geometrically as lines, planes, and generally speaking as linear manifolds. Hence, the detection of linear manifolds is a means by which correlations or linear dependencies may be identified. Correlations will further be discussed throughout chapters 9-11.

- Dimensionality is a fundamental challenge in many machine learning, pattern recognition, and data mining applications. This is primarily due to the "curse of dimensionality" discussed in section 5.1. The aim of dimensionality reduction is to reduce or transform high dimensional data into lower dimensional data while retaining most of the underlying structure in the data, and by that circumvent the problems associated with high dimensional data. Dimensionality reduction is also used to visualize high dimensional data in a lower dimensional space and by that gain more insight to the problem at hand. Very often observed real data is a consequence of a process governed by a small number of factors. In the data space this is manifested by the data points lying or being located close to surfaces such as linear manifolds whose intrinsic dimensionality is much smaller than the dimensionality of the data. Most dimensionality reduction techniques are "global" in the sense that they assume the whole data set can

be characterized by one lower dimensional surface, and overlook the possibility that different portions of the data lie on different lower dimensional surfaces, or might be generated by a process consisting of combination of simpler processes, i.e., a mixture of models. From a dimensionality reduction point of view the linear manifold paradigm of clustering can be thought of as a dimensionality reduction technique that assumes that the data lies on different linear surfaces whose intrinsic dimensionality is much smaller than that of the data.

Chapter 8

The Linear Manifold Clustering Algorithm

8.1 Overview

The linear manifold clustering algorithm proposed in this thesis is essentially a model-based clustering method that is based and supported by the model described in section 7.1. However, unlike typical model-based approaches such as the EM algorithm [31] which cast the clustering problem to a problem of determining the model parameters via an iterative optimization scheme, the algorithm proposed here is based on a stochastic-model-fitting approach that seeks to identify groups of points that may fit linear manifolds of various dimensionalities as defined by eq. (7.1.1).

The algorithm called LMCLUS (Linear Manifold CLUStering) can be viewed as an hierarchical-divisive clustering procedure. It executes three levels of iteration outlined in Fig. 8.1, and expects three inputs: L, an upper limit on the dimension of the linear manifolds in which we believe clusters may be embedded; S, a sampling level parameter used to determine the number of trial linear manifolds of a given dimensionality that will be examined in order to reveal the best possible partitioning of a given set of points; Γ, a sensitivity or "goodness of separation" threshold, which

is used to determine whether or not a partitioning should take place based on a trial linear manifold.

At the highest level of iteration the algorithm monitors the size of the data which is being partitioned. When no data is left to be partitioned the algorithm terminates. The second level of iteration causes the algorithm to iterate over a range of manifold dimensionalities, commencing with one-dimensional manifolds, and terminating with L-dimensional manifolds, where L is an input parameter. For each linear manifold dimension the algorithm enters the third level of iteration, in which *FindSeparation* outlined in Fig. 8.2 is invoked in an attempt to reveal separations among subsets of the data and to determine whether some of the points are embedded in linear manifolds. The idea behind *FindSeparation* is to successively sample points that can define a linear manifold of a given dimension, and select the linear manifold that is closest to a substantial number of points. This subset of closest points will typically correspond to a cluster. The proximity of the input data points to the manifold is captured by a distance histogram. If the manifold indeed has some subset of points near it, then the distance histogram will have a mixture of two distributions. One of the distributions has a mode near zero and is the distribution of distances of points belonging to a cluster. The other distribution is the distribution of the remaining points. The problem of separating the cluster points from the rest is then cast into a histogram thresholding problem. Upon termination *FindSeparation* returns four values γ- which is a measure of the "goodness" of the separation, τ- a proximity threshold that is computed from the histogram and is used to split the data into two groups, β- the basis of the manifold which exposes the separation, and ø-a point on the manifold representing its origin. When γ exceeds the value of the input sensitivity

threshold parameter Γ, indicating that a worthy separation has been found, then the data set is split according to τ. This split corresponds to the partitioning of all the points which are located close enough to the just determined manifold, i.e. all points that potentially belong to a given cluster, and those that belong to other clusters. In addition the dimension of the manifold which revealed the separation is recorded by assigning its value to $LmDim$. The third iteration continues reapplying *FindSeparation* in an attempt to further partition the cluster which may consist of sub-clusters, until the selected data points can not be further separated. At this point the algorithm will retract to the second level of iteration in an attempt to partition the cluster in higher dimensions, a process which will continue until the dimension limit L is reached. When L is reached we have a subset of the points that cannot be partitioned any more, and declare that a cluster is found. This cluster is labeled and added to the set of found clusters along with its dimensionality recorded by $LmDim$. The algorithm then retracts to the first level of iteration and is reapplied on the remaining set of points until no more points are left to be partitioned, detecting one cluster at a time. We note that if outliers exist then the last cluster/partition that is found will contain this set of points. By definition outliers do not belong to any cluster and therefore will remain the last group of points to be associated to any other group. Since they are unlikely to form any clusters the algorithm will not be able to partition them, and they will therefore be all grouped together.

8.2 Finding Separations

Let $C \subseteq X$ be a set of points that constitute a linear manifold cluster, and $C' = X - C$ be the set of points which do not belong to C. Assuming that C and C' do not

Algorithm LMCLUS $(X,\ L,\ S,\ \Gamma)$
$\mathcal{C} = \emptyset$ *# set of labeled clusters initially empty.*
$Dims = \emptyset$ *# set of intrinsic dimensionalities of each cluster.*
$i = 1$ *# cluster label*
while $X \neq \emptyset$ **do**
$Y = X,\ LmDim = 1$
for $k = 1$ **to** L **do**
while $[\gamma,\ \tau,\ \emptyset,\ \beta] =$ **FindSeparation**$(Y,\ k,\ S),\ \gamma > \Gamma$ **do**
 # a separation is revealed by a k-dimensional
 # manifold. Collect all points residing in the vicinity
# of that manifold.
$Y = \{x |\ x \in Y, \| (x - \emptyset) \|^2 - \| \beta'(x - \emptyset) \|^2 < \tau\}$
$LmDim = k$
 # a cluster is found, add it to the list of labeled clusters.
$C_i = Y,\ \mathcal{C} = \mathcal{C} \cup \{C_i\}$
 # recored the intrinsic dimensionality of the cluster
$dim_i = LmDim,\ Dims = Dim \cup \{dim_i\}$
$i = i + 1$
 # remove cluster points from the dataset.
$X = X - Y$
return $[\mathcal{C},\ Dims]$

Figure 8.1: The linear manifold clustering algorithm.

Algorithm FindSeparation $(X,\ k,\ S)$
$\gamma = -\infty,\ \tau = -\infty,\ \emptyset = \mathbf{0},\ \beta = \mathbf{0}$
$N = \min(\log \epsilon / \log(1 - (1/S)^k), c|X|)$
for $i = 1$ **to** N **do**
$M =$ **Sample** $(k+1)$ *points from* X
$O = x \in M$
$B =$ **FormOrthonormalBasis**$(M,\ O)$
$Distances = \emptyset$
for *each* $x \in X - M$ **do**
$y = x - O$
$Distances = Distances \cup \{\| y \|^2 - \| B'y \|^2\}$
$H =$ **MakeHistogram**$(Distances)$
$T =$ **FindMinimumErrorThreshold**(H)
$G =$ **EvaluateGoodnessOfSeparation**$(T,\ H)$
if $G > \gamma$ **then**
$\gamma = G,\ \tau = T,\ \emptyset = O,\ \beta = B$
return $[\gamma,\ \tau,\ \emptyset,\ \beta]$

Figure 8.2: Detecting separations among clusters embedded in lower dimensionality linear manifolds.

completely overlap, and that all points belonging to C fit the model given in equation 7.1.1, the goal is to separate C from C' and estimate the linear manifold in which C is embedded.

The distance of a point $\mathbf{x} \in X$ to a linear manifold defined by μ and the column vectors of B is given by

$$||(I - BB')(\mathbf{x} - \boldsymbol{\mu})|| = ||\overline{BB}'(\mathbf{x} - \boldsymbol{\mu})||. \tag{8.2.1}$$

As mentioned earlier in section 7.1 the points of C are likely to form a compact and dense region in the space orthogonal to the manifold in which they are embedded. Therefore by projecting X into the space spanned the column vectors of $\overline{B}$ and executing some form of clustering in the reduced space it is possible to identify and separate C from the rest of the data. However, eq. (8.2.1) shows that the distance of a point to the cluster center in the reduced space is equivalent to the distance of a

point to the linear manifold. Thus, rather than clustering in the reduced space it is also possible to measure distances from the manifold and collect all the points that lie in the vicinity of this manifold, essentially executing one-dimensional clustering. Since we are interested in estimating B, and because we are interested in detecting one cluster at a time, and since one-dimensional clustering is typically faster than clustering in higher dimensions, we choose to take this path.

Lemma 8.2.1. $||(I - BB')(\mathbf{x} - \boldsymbol{\mu})|| = \sqrt{||(\mathbf{x} - \boldsymbol{\mu})||^2 - ||B'(\mathbf{x} - \boldsymbol{\mu})||^2}$

Proof: Let $\mathbf{y} = \mathbf{x} - \mu$,

$$\begin{aligned}
||(I - BB')\mathbf{y}||^2 &= ||\mathbf{y} - BB'\mathbf{y}||^2 \\
&= (\mathbf{y} - BB'\mathbf{y})'(\mathbf{y} - BB'\mathbf{y}) \\
&= \mathbf{y}'\mathbf{y} - 2\mathbf{y}'BB'\mathbf{y} - \mathbf{y}'(BB')^2\mathbf{y} \\
&= \mathbf{y}'\mathbf{y} - \mathbf{y}'BB'\mathbf{y} \qquad (\ BB' \text{ is idempotent so } (BB')^2 = BB'\ ) \\
&= ||\mathbf{y}||^2 - ||B'\mathbf{y}||^2. \quad \square
\end{aligned}$$

Lemma 8.2.1 provides us a much more efficient way of computing the distance of a point to a manifold. If d is the dimension of the data, then computing the distance using lemma 8.2.1 gives us a speedup of $O(d)$, which for high dimensional data becomes a significant factor. To simplify the computation even further we choose to use the squared distance rather than the distance, henceforth we will use the term "distance" to mean the squared distance.

8.3 Minimum Error Thresholding

The problem of separating all the points that lie in the vicinity of a manifold can be cast into the problem of finding a *minimum error threshold* that is used to classify

points as either embedded in the manifold (belonging to C) or not, based on their distances to the manifold. Kittler and Illingworth [76] (KI) describe an efficient method for finding the minimum error threshold. Their method was designed for segmenting an object from its background in gray scale images using a grey level histogram of the image. Their method views the histogram segmentation problem as a two class classification problem, where the goal is to minimize the number of misclassified pixels. Analogous to our problem, the distance histogram can be viewed as an estimate of the probability density function $p(\delta)$ of the mixture population comprising of distances (δ) of points belonging to a linear manifold cluster and those that do not.

The KI procedure is based on the assumption that each component of the mixture is normally distributed. To support this assumption in our case, we note that as a consequence to the central limit theorem, the distances to the manifold, which are merely sums of random variables, will approach distribution-wise the normal, as the dimension of the space increases. Additional support was given by Diaconis and Freedman [77], who showed that most projections of high dimensional data to lower dimensions will be approximately normal.

Let δ be the distance of a point to the manifold, $p(\delta|i)$ be the probability density function of the distances of class i, where $i \in \{1, 2\}$, μ_i, σ_i be the mean and standard deviation of distances in class i, and $P(i)$ be the prior of class i. Then because of the normality assumption

$$p(\delta|i) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(\frac{-(\delta - \mu_i)^2}{2\sigma_i^2}\right), \tag{8.3.1}$$

and

$$p(\delta) = \sum_{i=1}^{2} P(i)p(\delta|i). \tag{8.3.2}$$

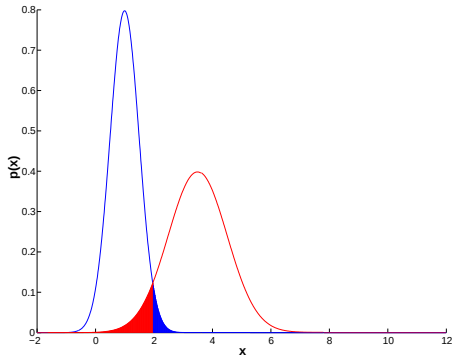


Figure 8.3: The classification error associated with a mixture of two Normal distributions is given by the area of the shaded regions of the two distributions.

Given $\mu_i, \sigma_i, P(i)$, and $p(\delta|i)$ there exists a threshold τ such that

$$P(1)p(\delta|1) > P(2)p(\delta|2) \quad \text{if} \quad \delta \leq \tau,$$

and

$$P(1)p(\delta|1) < P(1)p(\delta|2) \quad \text{if} \quad \delta > \tau,$$

where τ is the *Bayes minimum error threshold* [5]. As shown in figure 8.3, the minimum error threshold can be found by solving for δ the following equation

$$P(1)\frac{1}{\sqrt{2\pi}\sigma_1}\exp\left(\frac{(\delta-\mu_1)^2}{-2\sigma_1^2}\right) = P(2)\frac{1}{\sqrt{2\pi}\sigma_2}\exp\left(\frac{(\delta-\mu_2)^2}{-2\sigma_2^2}\right). \tag{8.3.3}$$

However, the true values of μ_i, σ_i, P_i are usually unknown. KI propose to obtain these estimates from the distance histogram h. Suppose that the histogram is thresholded at an arbitrary threshold t, then we can model the two resulting populations by a normal density $h(\delta|i,t)$ with parameters:

$$P_i(t) = \sum_{\delta=a}^{b} h(\delta), \tag{8.3.4}$$

$$\mu_i(t) = \frac{\sum_{\delta=a}^{b} \delta * h(\delta)}{P_i(t)}, \tag{8.3.5}$$

and

$$\sigma_i^2(t) = \frac{\sum_{\delta=a}^{b} (\delta - \mu_i(t))^2 * h(\delta)}{P_i(t)}, \tag{8.3.6}$$

where $a = 0$ and $b = t$ if $i = 1$, and $a = t + 1$ and $b = max(\delta)$ if $i = 2$. Now using the model $h(\delta|i, t)$ for $i \in \{1, 2\}$, the conditional probability of δ being correctly classified is given by

$$p(i|\delta, t) = \frac{h(\delta|i, t)P_i(t)}{h(\delta)}. \tag{8.3.7}$$

We wish to find the threshold t that maximizes this probability. Since $h(\delta)$ is independent of i and t it can be safely ignored. Furthermore, since the logarithm is a strictly increasing function, taking the logarithm and multiplying by a constant will not change the maximizing value. Therefore

$$\epsilon(\delta, t) = \left(\frac{\delta - \mu_i(t)}{\sigma_i(t)}\right)^2 + 2\log\sigma_i(t) - 2\log P_i(t) \tag{8.3.8}$$

can be considered as an alternative index of the correct classification performance. The overall performance is then given by the criterion function

$$J(t) = \sum_{\delta} h(\delta)\epsilon(\delta, t), \tag{8.3.9}$$

which reflects indirectly the amount of overlap between the two Gaussian populations. Substituting $\mu_i(t),\ \sigma_i(t),\ P_i(t)$, and $\epsilon(\delta, t)$ into $J(t)$ we get

$$J(t) = 1 + 2\left(P_1(t)\log\sigma_1(t) + P_2(t)\log\sigma_2(t)\right) - 2\left(P_1(t)\log P_1(t) + P_2(t)\log P_2(t)\right), \tag{8.3.10}$$

and the minimum error threshold selection problem can be formulated as

$$\tau = \arg\min_t J(t).$$

$J(t)$ can be computed easily and finding its minima is a relatively simple task as the function is smooth. We note that the tails of the distribution have been truncated by the thresholding operation and therefore the models $h(\delta|i,t)$, $i \in \{1,2\}$ will be biased estimates of the true mixture components. Cho, Haralick and Yi [78] proposed an improvement of KI's criterion function that corrects the biased variance estimates.

8.4 Sampling Linear Manifolds

A line, which is a 1D linear manifold, can be defined by two points, a plane which is a 2D manifold can be defined using three points. To construct a random k-dimensional linear manifold by sampling points from the data we need to sample $k+1$ linearly independent points. Let $\mathbf{x}_0, \ldots \mathbf{x}_k$ denote these points. Choosing one of the points, say $\mathbf{x}_0$ as the origin, the k vectors spanning the manifold are given by

$$\mathbf{y}_i = \mathbf{x}_i - \mathbf{x}_0$$

where $i = 1 \ldots k$. Fig. 8.4 illustrates this idea. The figure shows how a 2D linear manifold can be constructed by sampling three points, which are in turn used to define the two basis vectors spanning the manifold.

Assuming each of these sampled points came from the same cluster, then according to eq. (7.1.1)

$$\mathbf{y}_i = \mathbf{x}_i - \mathbf{x}_0 = (\boldsymbol{\mu}_0 + B\boldsymbol{\lambda}_i + \overline{B}\boldsymbol{\psi}_i) - (\boldsymbol{\mu}_0 + B\boldsymbol{\lambda}_0 + \overline{B}\boldsymbol{\psi}_0)$$

$$= B(\boldsymbol{\lambda}_i - \boldsymbol{\lambda}_0) + \overline{B}(\boldsymbol{\psi}_i - \boldsymbol{\psi}_0).$$

If the cluster points did not have an error component off the manifold, i.e., they all lie on the linear manifold, then sampling any $k+1$ points which are linearly independent

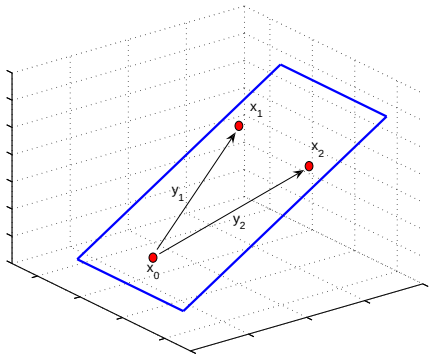


Figure 8.4: A 2D linear manifold defined by three points.

and belong to the same cluster would enable us to reconstruct B. So in order to get a good approximation of B we would like each of the sampled points to come from the same cluster and to be as close as possible to the linear manifold. From the equation above we see that this occurs when

$$\boldsymbol{\psi}_i - \boldsymbol{\psi}_0 \approx \mathbf{0},$$

resulting in a set of k vectors which are approximately a linear combination of the original vectors in B. A good indication as to why this is likely to occur when the sampled points come from the same cluster, is given by the fact that

$$\mathrm{E}[\boldsymbol{\psi}_i - \boldsymbol{\psi}_0] = \mathbf{0},$$

and that normally distributed data ($\boldsymbol{\psi}_i - \boldsymbol{\psi}_0$ follows a normal distribution) tends to cluster around its mean. In cases where the clusters are well separated, the requirement that $\boldsymbol{\psi}_i - \boldsymbol{\psi}_0 \approx \mathbf{0}$ can be relaxed. That is, when the clusters are well separated more sets of points coming from the same cluster, and not only those that are relatively close to the manifold will be good candidates to a construct a manifold that

will induce a large valley in the distance histogram that separates the linear manifold cluster from the remaining points. As a consequence, the problem of sampling a linear manifold that will enable us to separate a linear manifold cluster from the rest of the data can be reduced to the problem of sampling $k+1$ points that all come from the same cluster.

Assuming there are S clusters in the data set whose size is distributed with low variance, then for large data sets the probability that a sample of $k+1$ points all come from the same cluster is approximately

$$\left(\frac{1}{S}\right)^k. \tag{8.4.1}$$

If we want to ensure with probability $(1-\epsilon)$ that at least one of our random samples of $k+1$ points all come from the same cluster, then we should expect to make at least n selections of $k+1$ points, where

$$\left[1-\left(\frac{1}{S}\right)^k\right]^n \leq \epsilon, \tag{8.4.2}$$

yielding that

$$n \geq \frac{\log\ \epsilon}{\log(1-(1/S)^k)}. \tag{8.4.3}$$

Therefore, by computing n given ϵ, and S which is an input to the algorithm we can approximate a lower bound on the number of samples required, such that with high probability, at least one of the n samples contains $k+1$ points that all come from the same cluster. Unlike algorithms that require the number of clusters as input such as K-Means, the user input S does not predetermine the number of clusters LMCLUS will output. It is a rough estimate of the number of clusters in the data set, which is only used to compute an initial estimate of the sample size required to ensure we

sample points coming from the same cluster. It is rather a "gauge" with which we can tradeoff accuracy with efficiency.

The sample size n grows exponentially with the dimensionality of the linear manifold k which is being sampled. Although LMCLUS is not designed to identify clusters embedded in linear manifolds of high dimension, it is possible to bound the sample size from above. Dasgupta and Gupta [79] showed that using random projections it is possible to project high dimensional data into a low dimension subspace and preserve distances up to a factor of $1+\epsilon$ with probability $1/|X|$, where $|X|$ is the size of the data set. This in turn means that by performing $O(|X|)$ random projections it is possible to find a projection that will be able to retain a good level of separation between clusters with high probability. Combining this result with the sample size computed in eq. (8.4.3) the number of samples N which will be drawn can by determined by the following heuristic

$$N = \min\left(\log\epsilon / \log(1-(1/S)^k), c|X| \right), \tag{8.4.4}$$

where c is some constant independent of $|X|$ which can simply be set to equal one, making $|X|$-the size of the data being clustered an alternative sample size approximation. We note that for small k and large data sets the sample size computed by eq. (8.4.3) is likely to be much smaller than $|X|$, and will therefore be preferred.

8.5 Putting it All Together

For each sample of points $\{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$ we construct an orthonormal basis B of a linear manifold using the Gram-Schmidt process. (If the Gram-Schmidt process indicates that the sampled points are not linearly independent, a new sample of points is taken.) Then using KI's method we compute a threshold τ. Of all possible thresholds

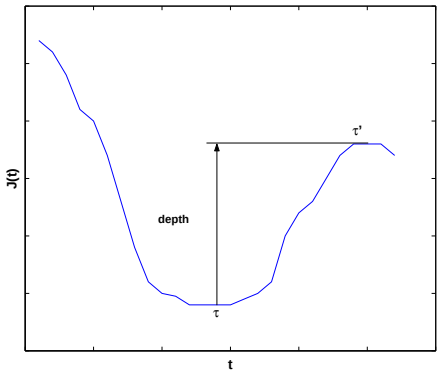


Figure 8.5: The depth of separation .

corresponding to different linear manifolds which induce different separations, we prefer the one which induces the best separation. The best separation is defined as the separation which induces the largest discriminability given by

$$discriminability = \frac{(\mu_1(\tau) - \mu_2(\tau))^2}{\sigma_1(\tau)^2 + \sigma_2(\tau)^2}, \tag{8.5.1}$$

and the one which causes the deepest broadest minimum in KI's criterion function- $J(t)$. The deepest minimum can be quantified by computing the difference/depth of the criterion function evaluated at the minimum τ and the value evaluated at the closest local maxima τ' as illustrated in Fig. 8.5, and given by

$$depth = J(\tau') - J(\tau). \tag{8.5.2}$$

The composite measure of the goodness of a separation is then given by

$$G = discriminability \times depth. \tag{8.5.3}$$

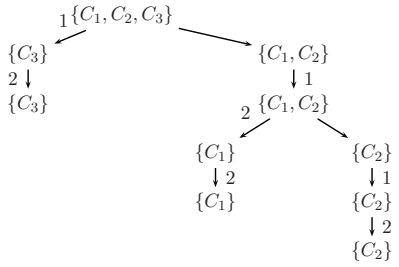
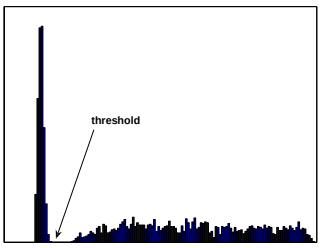
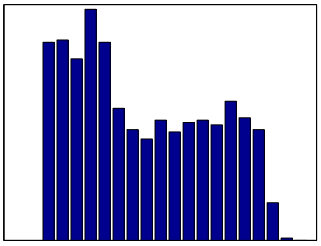


Figure 8.6: A tree summarizing the clustering process of the sample data set from Fig. 7.1. The labels on the arrows specify the dimensionality of the linear manifold which was used to separate the clusters.

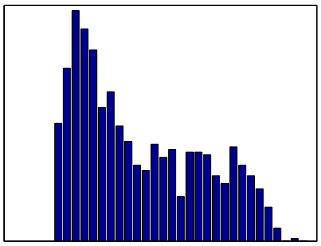
A typical run of the algorithm is illustrated in Figs. 8.6 and 8.7 by a tree which summarizes the clustering process of the sample data set depicted in Fig. 7.1, and the corresponding histograms that were produced by the clustering process and used to separate the linear manifold clusters in the data set. At the beginning of the process the algorithm searches for one-dimensional linear manifolds in which some clusters may be embedded. Since cluster C_3 is such a cluster it is separated from C_1, C_2 using the threshold returned by KI's procedure, and the algorithm proceeds by trying to further partition it in higher dimensions. Since it cannot be further partitioned using two-dimensional manifolds, cluster C_3 is declared to be found. The algorithm then attempts to separate the remaining clusters C_1, C_2 using one-dimensional manifolds. Since both these clusters are embedded in two-dimensional linear manifolds the algorithm will fail. However by trying to separate them using 2D manifolds the algorithm will succeed. At this point the algorithm will attempt to further partition each of the clusters C_1 and C_2, however since they are inseparable C_1 and C_2 are declared to be found, and the algorithm terminates.



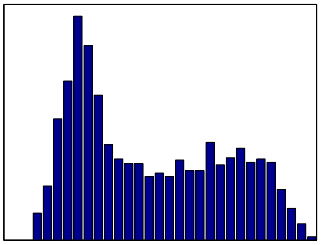
(a) C_3 is separated from C_1 and C_2 at D1



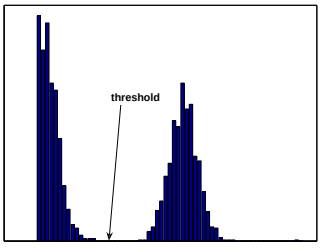
(b) C_3 cannot be separated at D1



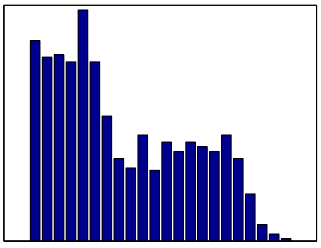
(c) C_3 cannot be separated at D2, C_3 is a cluster



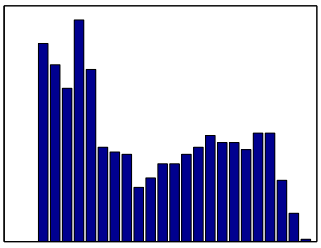
(d) C_1 and C_2 cannot be separated at D1



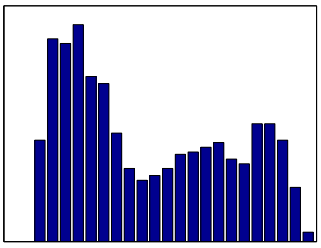
(e) C_1 is separated from C_2 at D2



(f) C_1 cannot be separated at D2, C_1 is a cluster



(g) C_2 cannot be separated at D1



(h) C_2 cannot be separated at D2, C_2 is a cluster

Figure 8.7: Histograms produced by the clustering process and used to separate the linear manifold clusters of Fig. 7.1.

8.6 Algorithm Speedup by Histogram Sampling

Rather than using all the points that are to be partitioned to construct distance histograms on which KI's technique is applied, statistical sampling theory suggests that it is possible to sample a smaller representative subset of the points. That is, it is possible to construct representative histograms using fewer points, which capture the essential properties of the two populations we are interested in separating. Preserving these properties will enable us to obtain a sufficiently accurate threshold that can be used to separate the two populations, when such a separation exists. The properties we seek to preserve in the sample are two:

1. We would like the proportion of sampled points coming from each of the populations of the mixture to be similar to the proportion in the unsampled mixture. In other words, we do not want the majority or all of the sampled points to come from one of the populations, whereas the other is represented by very few or no points.

2. We would like the sample means of the two populations to be similar to the means of the two populations in the unsampled mixture. This ensures that we will not be sampling too many points that come from the tails of the original distributions.

Let N be the total number of points to be partitioned, p be the fraction of points coming from the left population, i.e. the points that potentially belong to a cluster, and n_p be the size of a sample to be drawn. Since it makes very little difference for large populations whether we sample with or without replacement, the distribution of the number of points coming from the cluster given a sample of size n_p, can be

modeled by binomial distribution with parameters n_p and p. If $\hat{p}$ is the actual fraction of the points in our sample that come from the cluster, then we would like to choose n_p, such that with relative error $\delta_p p$ and with confidence $1-\epsilon$, where $0 \leq \delta_p, \epsilon \leq 1$

$$P[|\hat{p} - p| \leq \delta_p p] \geq 1 - \epsilon. \tag{8.6.1}$$

Due to the the *central limit theorem* when n_p is large, which is likely to be our case, and due to stringent requirements on δ_p and ϵ, the Normal distribution can be used as an approximation to construct a confidence interval for a binomial random variable, where $\mathrm{E}[\hat{p}] = p$ and $\mathrm{Var}[\hat{p}] = p(1-p)/n_p$. Hence,

$$P[|\hat{p} - p| \leq \delta_p p] = P\left[\frac{-\delta_p p}{\sqrt{\frac{p(1-p)}{n_p}}} \leq \frac{\hat{p} - p}{\sqrt{\frac{p(1-p)}{n_p}}} \leq \frac{\delta_p p}{\sqrt{\frac{p(1-p)}{n_p}}}\right] \geq 1 - \epsilon, \tag{8.6.2}$$

where $\frac{\hat{p}-p}{\sqrt{\frac{p(1-p)}{n_p}}}$ follows a standard Normal random variable Z. According to eq. (8.6.2), to estimate n_p given ϵ, δ_p we need to ensure that

$$P\left[Z \geq \frac{\delta_p p}{\sqrt{\frac{p(1-p)}{n_p}}}\right] \leq \frac{\epsilon}{2}, \tag{8.6.3}$$

yielding that

$$n_p \geq \left(\frac{1-p}{p}\right) \frac{Z^2_{\epsilon/2}}{\delta_p^2}, \tag{8.6.4}$$

where $Z_{\epsilon/2}$ is the value for which the probability that a standard Normal random variable is greater than $\epsilon/2$, e.g., $Z_{0.025} = 1.96$. However n_p cannot be computed without knowing p which is unknown. Since $\hat{p}$ is a *consistent estimator* of p, i.e., the larger the sample is the more likely we are to get a better estimate of p, we propose to gradually increase the size of the sample and obtain n_p as follows: we initialize p to some value p_0 which could be a function of the input parameter S, such as $p_0 = 1/S$. Given p_0 we then sample n_{p_0} points with which we create an initial

distance histogram. Using this histogram we obtain a threshold which induces a new estimate p_1. If $n_{p_1} \leq n_{p_0}$ we stop and use p_0 as our estimate of p and n_{p_0} as our sample size n_p, if not indicating we need to sample more points (because $p_1 < p_0$), we repeat the same procedure. That is, using a sample of size n_{p_i} we obtain a new estimate p_{i+1} with which a new $n_{p_{i+1}}$ is computed until for some k $n_{p_{k+1}} \leq n_{p_k}$. This is likely to occur after a small number of iterations. Note that the computational overhead is never going to be larger than if we were to use no sampling, i.e., use all the points to construct histograms.

Next we need to compute a sample size n_μ that is needed to satisfy the second requirement. That is, the sample size that is required to ensure with high probability that the means of the two populations in the sample will not deviate significantly from the unsampled means of the two populations. More formally, if $\bar{x}$ is the sample mean of one of the populations, μ the unsampled mean of that population, σ^2 the variance of that population, p the fraction of points coming from the population which we estimated previously, δ_μ the deviation from the true mean measured in units of σ, and $1 - \epsilon$ the confidence of our estimate, then we would like to find n_μ satisfying

$$P\left[|\bar{x} - \mu| \leq \delta_\mu \sigma\right] \geq 1 - \epsilon \tag{8.6.5}$$

for both populations. For the population whose fraction is p, eq. (8.6.5) can be rewritten as

$$P\left[-\delta_\mu \sqrt{n_\mu p} \leq \frac{\bar{x} - \mu}{\sigma/\sqrt{n_\mu p}} \leq \delta_\mu \sqrt{n_\mu p}\right] \geq 1 - \epsilon. \tag{8.6.6}$$

Since $\frac{\bar{x}-\mu}{\sigma/\sqrt{n_\mu p}}$ follows a standard Normal random variable, $n_\mu p$ can be estimated similar to n_p by

$$n_\mu p \geq \frac{Z^2_{\epsilon/2}}{\delta^2_\mu}, \tag{8.6.7}$$

and similarly for the other population

$$n_\mu(1-p) = \frac{Z^2_{\epsilon/2}}{\delta^2_\mu}. \tag{8.6.8}$$

Therefore, the total number of samples n_μ needed to satisfy the second requirement for both populations is

$$n_\mu = \frac{Z^2_{\epsilon/2}}{\delta^2_\mu \ \min\{p, 1-p\}}, \tag{8.6.9}$$

and the total number of samples needed to satisfy both requirements is determined by the following heuristic

$$\min\left\{|N| \ , \ \max\left\{\left(\frac{1-p}{p}\right)\frac{Z^2_{\epsilon/2}}{\delta^2_p} \ , \ \frac{Z^2_{\epsilon/2}}{\delta^2_\mu \min\{p, 1-p\}}\right\}\right\}. \tag{8.6.10}$$

8.7 Complexity Analysis

In this section we present asymptomatic analysis of LMCLUS's running time in terms of the number of data points, the dimensionality of the data, the highest dimensionality of the linear manifolds in which clusters are embedded, and the number of clusters which are detected. Because LMCLUS uses sampling, complexity analysis of its running time is not straightforward. As a consequence the following result is only a worst case upper bound on its running time, and does not reflect its performance in practice as demonstrated by the experiments presented in section 8.8.

Lemma 8.7.1. *The overall worst case time complexity (an upper bound) of LMCLUS in terms of the size of the data-N, the dimensionality of the data-d, the largest dimension of the linear manifolds in which clusters are embedded-L, and the number of clusters which are detected-K, is* $O(N^2K^2L^3d)$.

Proof: The overall complexity of LMCLUS depends the complexity of the *FindSeparation* procedure, and on the number of calls to it. Starting with the *FindSeparation* procedure, each set of $k+1$ points that are sampled in order to construct a

k-dimensional manifold must first be transformed to a set of k orthonormal vectors (*FormOrthonormalBasis*) by a Gram-Schmidt process whose complexity is $O(k^2 d)$, where d is the dimensionality of the data. Measuring distances to the manifold amounts to a projection whose complexity is $O(kd)$. Assuming there are n points in data, the overall complexity of this step is $O(nkd)$. Making a distances histogram requires $O(n)$ operations (assigning each distance to a bin). Assuming the number of bins in the histogram is bound by some constant, then finding the minimum error threshold by Kittler and Illingworth's method has complexity $O(1)$ (constant time). This is because computing the mean and standard deviation of each population for each threshold t can be done in time that is linear in the number of bins. Likewise we only need to compute $J(t)$-the minimum error index function, for a constant number of different bins. Evaluating the goodness of a separation involves only one multiplication of two values. Each of the steps described above must be repeated for each linear manifold that is sampled. Since the number of linear manifolds sampled is bounded from above by the number of points that are to be clustered (eq. (8.4.4)), the overall complexity of *FindSeparation* is

$$O(nk^2 d + n^2 kd + n^2 + n).$$

We now need to determine the number of calls to the *FindSeparation* procedure. Assuming the clusters are separable, then in the worst case each cluster is found at the highest dimension which is L, that is, up to dimension $K-1$ the data is inseparable, and at dimension L, a cluster is found only after repeated calls to *FindSeparation* that gradually peals off other clusters one at a time. If there are K clusters in the data, of approximately the same size, then the overall complexity of LMCLUS as a

function of N, K, L, d is given by:

$$\begin{aligned} & \sum_{k=1}^{K} \left[\sum_{l=1}^{L-1} \frac{kN}{K} \left(l^2 d + \frac{kNld}{K} + \frac{kN}{K} + 1 \right) + \sum_{i=1}^{k} \frac{iN}{K} \left(L^2 d + \frac{iNLd}{K} + \frac{iN}{K} + 1 \right) \right] \\ = \; & O(N^2 K^2 L^3 d). \quad \square \end{aligned} \tag{8.7.1}$$

This result shows that LMCLUS's running time is only linear in the dimensionality of the data and cubic in the dimensionality of the linear manifolds in which clusters are embedded. Related methods such as ORCLUS [54] are cubic in the dimensionality of the data due to the computation of principle components, or like CLIQUE [28], exponential in the dimension of subspaces in which clusters are embedded. We also note that although the worst case complexity of LMCLUS is quadratic in the number of points, in practice when the dimension of the linear manifolds is much smaller than the size of the data set, the sample size computed by eq. (8.4.4) will be much smaller than size of the data, resulting in an algorithm whose complexity is only linear in the size of the data. The histogram sampling optimization further enhances this result.

8.8 Experiments with Synthetic Data

In this section we present a broad evaluation of LMCLUS applied on synthetic data sets. LMCLUS as well as three other related methods: DBSCAN [18] a representative of the full-dimensional clustering methods, ORCLUS [54] a representative of subspace clustering methods, and HPPC [80] a projection pursuit based clustering method that repeatedly bi-partitions a data set by searching for separations in one-dimensional projections of the data, were implemented in C++. The experiments were performed on a Linux based workstation with an Intel P4 3.0Ghz CPU and 1GB of memory. The aim of the experiment was to evaluate LMCLUS's performance in comparison to

the other methods with respect to accuracy, efficiency, scalability and its stability as a stochastic algorithm.

8.8.1 Synthetic Data Generation

Several dozen data sets were generated according to the linear manifold cluster model specified in eq. (7.1.1). The method used to generate the data sets was based on the method used in the ORCLUS paper. The underlying idea is to first generate each cluster in an axis-parallel manner centered at the origin, and then randomly translate and rotate it, where the rotation is used to achieve the effect of an arbitrary orientation of the cluster in the space. The translation and rotation correspond in the ORCLUS paper to the random "anchor points" chosen for each cluster, and to the "random matrix" which is used to determine the orientation of each cluster. In addition to the clusters, noise points are scattered uniformly in the space. In the following we give a detailed description of this process with which we generated two types of data sets differing only by the amount in which each cluster is translated.

Let K be the number of clusters, d be the dimensionality of the data set, N be the total number of points in the data set, N_{noise} be the number of noise or outlier points, and R be the range along each of the dimensions of the manifold. The number of points assigned to cluster i denoted by C_i is determined by $|C_i| = (N - N_{noise}) \cdot r_i / \sum_{i=1}^{K} r_i$, where r_i is a realization of an *exponential* random variable with mean 1. The dimensionality of the manifold in which C_i is embedded denoted by l_i is determined by a realization of a *Poisson* random variable with mean l, typically much smaller than d, and where l_i must satisfy $1 \leq l_i < d$.

The points for cluster C_i are generated as follows: l_i dimensions out of the d possible dimensions are uniformly selected as the dimensions of the linear manifold

in which C_i is embedded. In each of these dimensions the points are independently and identically distributed (i.i.d.) according $U(-R/2, +R/2)$. In the other $d - l_i$ dimensions representing the space orthogonal to the manifold and the perturbation off the manifold, the points are normally distributed with mean 0 and standard deviation $\sigma_{ij} = s \cdot \alpha_{ij}$ for the j-th dimension, where s is some constant and α_{ij} is a realization of a uniformly distributed random variable in the range $[1, 2]$. All the data sets generated as part of our evaluation used $R = 1000$, while varying K, d, N, N_{noise}, l, s.

After C_i is created in an axis parallel manner, the next step is to translate and rotate it. In order to translate the cluster we generate a random $d \times 1$ vector $\mathbf{T}_i$ whose entries are i.i.d. $U(-t/2, +t/2)$, where t is some constant which determines the type of data set generated. The smaller t is, the closer the clusters are likely to be to each other, and the more likely they are to overlap. Once $\mathbf{T}_i$ is obtained we add it to each point $\mathbf{x} \in C_i$, i.e., $\mathbf{x} = \mathbf{x} + \mathbf{T}_i$. In order to rotate C_i we perform a series of planar rotations by going through all pairs of distinct dimensions, each defining a different plane. Let (m, n) denote some pair. For each pair we generate a random angle θ_{mn} uniformly distributed in the range $[0, 2\pi]$. We then compute the (m, n) planar rotation for each $\mathbf{x} \in C_i$ by $x_m = x_m cos\theta_{mn} - x_n sin\theta_{mn}$ and $x_n = x_m sin\theta_{mn} + x_n cos\theta_{mn}$, where x_m, x_n are the m-th and n-th components of vector x.

The noise set was generated by distributing each of its points uniformly across all dimensions according to $U(-(R+t)/2, +(R+t)/2)$. The outcome of this process was a set of clusters all residing in an hypersphere of radius $(R + t)/2$.

Once a data set is generated we would like to have the ability to determine some of its characterizing properties. Specifically, we would like to devise a measure that would describe the relative sparsity of the clusters that were generated, and a measure

that is able to convey some indication as to the relative 'hardness' or difficulty that the generated data set might present to various clustering algorithms. Since LMCLUS is aimed at finding discrete partitions of the data we would also want this measure to give us some indication pertaining to the amount of overlap between the generated clusters. A candidate data set which we aimed at generating was one in which the clusters were relatively close in some subspace with minimal overlap, and yet sparse enough in the full space that canonical clustering algorithms would not be able to detect. The sample data of Fig. 7.1 is an example of such a data set. We used the *cluster sparsity coefficient* [54] proposed in the ORCLUS paper as a measure of the relative sparsity of the generated data sets, and indirectly as a measure of the relative "hardness" or difficulty a potential data set might present to a clustering algorithm. The authors propose to apply this measure ranging in $[0, 1]$ on the full space, and note that high values indicate that the data set generated is one on which full-dimensional or axis-parallel projected clustering would be meaningless. This measure is essentially a weighted average of the within cluster variances divided by the total data variance. More formally, let $\boldsymbol{\mu}_i$ be the mean of cluster i, $\boldsymbol{\mu}$ be the mean of the whole data set, C_i be the set of points in cluster i, X the whole data set, K the number of clusters in the data set, and $\mathbf{x}$ some point in X, then the cluster sparsity coefficient is defined by

$$\frac{1}{K}\sum_{i=1}^{K}\frac{1/|C_i|\sum_{\mathbf{x}\in C_i}\parallel \mathbf{x}-\boldsymbol{\mu}_i\parallel^2}{1/|X|\sum_{\mathbf{x}\in D}\parallel \mathbf{x}-\boldsymbol{\mu}\parallel^2}. \tag{8.8.1}$$

Obviously when this value is small (close to zero) we may assume that the clusters form compact clouds that are well separated from one another. However large values (close to one) do not only indicate that the data set or clusters are very sparse but could also suggest a significant overlap among the clusters. This is because clusters

that significantly overlap are likely to have centers which are close to each other, and therefore close to the center of the whole data set. As mentioned earlier significant cluster overlap is not a desired property for the the data sets we would like to produce.

Two families of data sets were generated as part of the evaluation. The first having cluster sparsity coefficient in the range $[0.5, 0.6]$, yielding data sets that we found to be relatively hard for full-dimensional clustering algorithms, yet with very small and possibly no cluster overlap. These type of data sets were generated by fixing $s = 20$ and setting t such that the cluster sparsity coefficient yielded a value in the range $[0.5, 0.6]$. The data set depicted in Fig. 7.1 for example has a cluster sparsity coefficient value of 0.54. The second type, which we call *star* data sets due to their star like geometry, yielded cluster sparsity coefficient values close to 0.99. The star data sets were generated so that cluster centers were relatively close to each other, yet with minimal overlap. This was done by assigning to t and s smaller values. Fig. 8.8 shows an example of a star type data set consisting of one-dimensional linear manifold clusters.

8.8.2 Parameter Setting

LMCLUS requires the setting of three input parameters: L, an upper limit on the dimension of the linear manifolds in which we believe clusters may be embedded; S, a sampling level parameter which based on the heuristic outlined in section 8.4; we suggest setting it to a rough estimate of number of clusters that might exist in the data set; Γ, a sensitivity or "goodness of separation" threshold, which the higher it is set the more coarse the clustering will be, and the lower it is set the finer (larger number of clusters and smaller clusters) the clustering will be. The first two input parameters are fairly intuitive and easy to set. For Γ we have yet to find a general

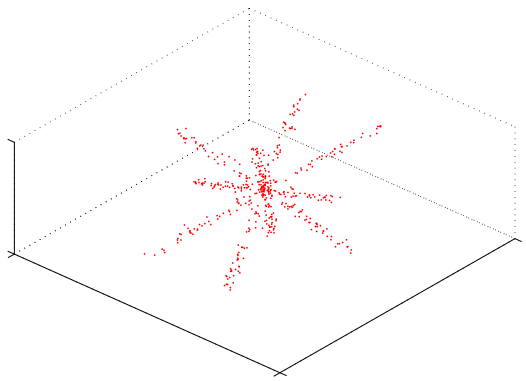


Figure 8.8: A 3D *star* type data set consisting of 1D linear manifold clusters.

approach, and similar to other methods, we suggest setting by experimenting with different values. Nonetheless, we found that not many trials are required until a reasonable clustering is obtained. Moreover, we found that when clusters are well separated setting $\Gamma = 1.0$ will yield good results and anything higher than 1.0 will only return the same results. I.e. $\Gamma = 1.0$ can be thought of as a threshold for the case of well separated clusters. When clusters are closer to each other or tend to overlap the value of Γ must be decreased. Based on many observations and experiments with data sets having clusters that are relatively close, we suggest setting Γ in the range $[0.4, 0.5]$. Any clustering obtained by setting $\Gamma < 0.4$ indicates significant cluster overlap. Another somewhat less experimental approach to obtain the correct value for Γ will be to record many goodness measures returned by the *FindSeparation* procedure on a given data set or a family of data sets, threshold them using some thresholding technique, and use the threshold as the value which is to be assigned to Γ.

8.8.3 Accuracy

From a pool of dozens of synthetic data sets on which LMCLUS was applied, a representative sample of fifteen was selected for illustrative purposes. Their characteristics are summarized in table 8.1, where the *star* data sets are marked with a star ($'*'$). As a cluster validity measure of *accuracy* we used *Cluster Purity* [81] operating on a *Confusion Matrix*. Confusion matrices are used to indicate how well the output clusters match with the input clusters, also referred to as "ground truth" in a supervised environment. The (i, j) entry of the matrix specifies the number of points belonging to output cluster labeled i, which were generated as part of input cluster labeled j. Ideally, when the clustering algorithm performs well, there should be a single dominant entry in each row that establishes a match of an output cluster to an input cluster. Cluster purity is then used to quantify this match and the overall quality of the clustering which we refer to as accuracy. Let K be the number of output clusters, $|X|$ the size of the data set, $|C_i|$ the size of output cluster i, and $|C_{ij}|$ the number of points output cluster i and input cluster j have in common. Then the purity of cluster C_i is defined as

$$Purity(C_i) = \frac{1}{|C_i|} \max_j(|C_{ij}|), \tag{8.8.2}$$

and overall purity or what we refer to as the overall accuracy of the clustering as

$$Purity = \sum_{i=1}^{K} \frac{|C_i|}{|X|} purity(C_i) = \frac{1}{|X|} \sum_{i=1}^{K} \max_j(|C_{ij}|), \tag{8.8.3}$$

which is essentially a weighted sum of individual cluster purities.

Table 8.2 summarizes the overall accuracy, along with the amount of time required by each algorithm we tested to cluster the sample of fifteen data sets. Apart from DBSCAN, each of the other algorithms due to their stochastic nature were run several

times and their averages were recorded. The results depicted in table 8.2 clearly demonstrate LMCLUS's superiority over the other clustering algorithms. LMCLUS was the only algorithm able to cluster with over 0.85 accuracy all the fifteen data sets. ORCLUS ranked second, was able to cluster accurately ten of the fifteen data sets. HPPC ranked third, clustered accurately eight of the fifteen data sets. DBSCAN ranked last, was able to cluster accurately only three of the data sets. The inability of DBSCAN to cluster these data sets is no surprise for clustering algorithms that utilize distance metrics in the full space to cluster subspace or linear manifold clusters. Only LMCLUS and ORCLUS were able to handle the *star* clusters. In some cases ORCLUS slightly outperformed LMCLUS by a small margin. The fact that HPPC was not able to cluster the *star* data sets also comes at no surprise. This is because as a projection pursuit algorithm HPPC searches for "interesting" separations revealing 1D projections, and any 1D projection of these type of data sets, due to the star like arrangement of clusters in the space, will only reveal "uninteresting" unimodal distributions of projections that cannot be used to separate the data. However its ability to cluster well the first type of data sets (8/10 as opposed to 5/10 which ORCLUS clustered accurately) supports the concept of random projections as a tool for detecting inter-cluster separations in high dimensional spaces. In terms of running time (overall time required to cluster all data sets), LMCLUS ranked second to HPPC. The remarkable low running times of HPPC can be attributed to the fact that it uses a stochastic "genetic" approach to select interesting 1D projections using a constant number of "generations", making it scale linearly with the size and dimensionality of the data. Nonetheless LMCLUS runs faster than the other algorithms on seven of the fifteen data sets, and when compared to ORCLUS and DBSCAN only, demonstrates

	data size	num of clusters	space dim	manifold dim
D_1	3000	3	4	2-3
D_2	3000	3	20	13-17
D_3	30000	4	30	1-4
D_4	6000	3	30	4-12
D_5	4000	3	100	2-3
D_6	90000	3	10	1-2
D_7	5000	4	10	2-6
D_8	10000	5	50	1-4
D_9	80000	8	30	2-7
D_{10}	5000	5	3	1-2
$*D_{11}$	1500	3	3	1
$*D_{12}$	1500	3	3	2
$*D_{13}$	1500	3	7	3
$*D_{14}$	5000	5	20	4
$*D_{15}$	4000	4	50	3

Table 8.1: A sample of 15 data sets used to evaluate the "Accuracy" of selected clustering algorithms. The table lists the characteristics of each of the data sets.

a significant gain in efficiency, especially when applied on large or high dimensional data sets.

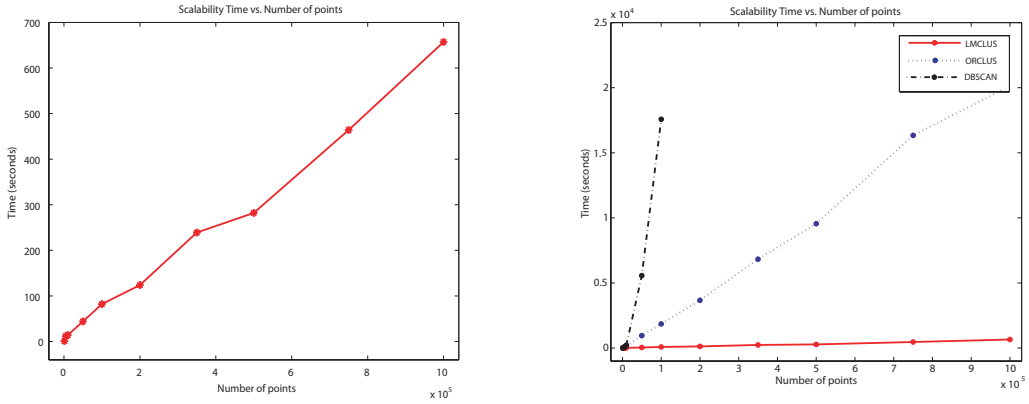
8.8.4 Scalability

Scalability was assessed in terms of the size and dimensionality of the data. In the first set of tests, we fixed the number of dimensions at 10, and the number of clusters to 3, each of same size, and embedded in a 3D manifold. We then increased the number of points from 1,000 to 1,000,000. In the second set of tests we fixed the number of points and clusters as before, but increased the number of dimensions in the data set from 10 to 120. Similar to the other experiments, due to their stochastic nature, LMCLUS and ORCLUS were run several times on each data set, and the average was taken as the reported running time. Figs 8.9 and 8.10 are plots of the running

	LMCLUS		ORCLUS		DBSCAN		HPPC	
	accuracy	time	accuracy	time	accuracy	time	accuracy	time
D_1	0.95	0:0:08	0.80	0:0:22	0.34	0:0:9	0.72	0:0:51
D_2	0.98	0:0:33	0.59	0:2:18	0.65	0:0:36	0.97	0:1:39
D_3	1.00	0:15:38	0.65	1:5:30	1.00	1:31:52	0.99	0:1:32
D_4	0.99	0:9:22	0.98	0:8:20	0.66	0:3:49	0.97	0:0:12
D_5	1.00	0:0:20	0.88	0:54:30	0.65	0:5:24	0.99	0:3:54
D_6	0.99	0:0:29	1.00	0:29:02	0.67	4:58:49	1.00	0:1:23
D_7	0.99	0:2:05	0.99	0:2:41	0.74	0:0:54	0.96	0:0:35
D_8	0.99	0:1:42	0.63	1:33:52	1.00	0:17:00	0.99	0:3:43
D_9	0.99	3:12:46	0.96	13:30:30	1.00	10:51:15	0.99	0:4:57
D_{10}	0.86	0:0:48	0.68	0:0:45	0.59	0:0:5	0.78	0:0:33
$*D_{11}$	0.98	0:0:01	0.99	0:0:10	0.43	0:0:02	0.33	0:0:52
$*D_{12}$	0.97	0:0:02	0.99	0:0:11	0.34	0:0:02	0.33	0:0:26
$*D_{13}$	0.97	0:0:05	0.99	0:0:17	0.33	0:0:04	0.33	0:0:34
$*D_{14}$	0.99	0:5:46	1.00	0:10:42	0.21	0:1:39	0.20	0:1:30
$*D_{15}$	0.99	0:9:14	1.00	0:25:52	0.25	0:2:34	0.25	0:3:20

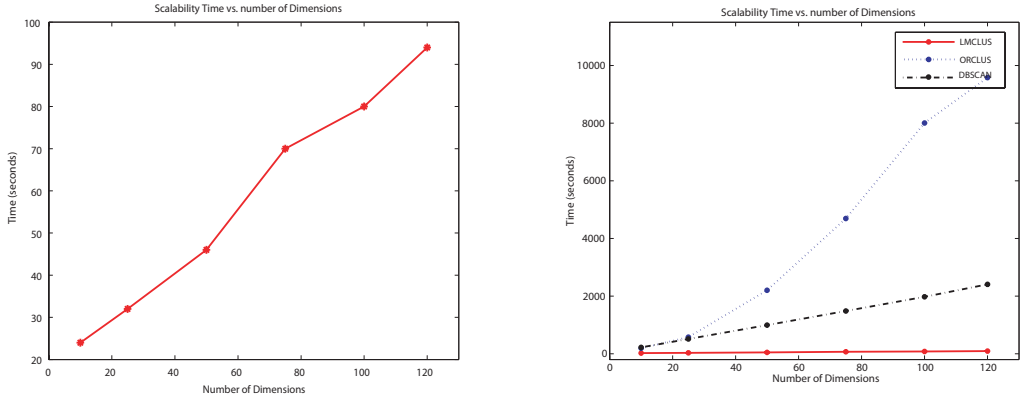
Table 8.2: Accuracy and running times (in hours, minutes, and seconds) comparison. Each algorithm was applied on a sample of fifteen data sets whose characteristics are listed in table 8.1.

time for the two sets of tests. Although LMCLUS's asymptotic time complexity indicates that it is quadratic in the size of the data, these set of tests confirm our conjecture from section 8.7 that in practice LMCLUS will scale much better. Fig. 8.9(a) shows that in practice, for data sets with a small number of clusters which are embedded in low dimensionality manifolds, LMCLUS like ORCLUS with complexity $O(K^3 + kNd + K^2d^3)$ scales linearly with the size of the data. This as mentioned before can be attributed to three properties inherent to the algorithm. The first is that each cluster that is detected is removed from the data set, thus shrinking the set of points considered for future iterations. The second, is due to histogram sampling which significantly reduces the amount of projections that need to be computed in order to construct histograms which yield the thresholds that are used to separate subsets of points. The third reason is that although the number of manifolds that are sampled in order to detect each cluster is bounded from above by the size of the data set, for lower dimensionality manifolds the number of manifolds that are actually constructed will be much lower and dominated by the heuristic given in eq. (8.4.3). We note however that as the dimensionality of the cluster manifolds increases, performance will degrade. Fig. 8.10(a) validates our analytical complexity analysis that LMCLUS, like DBSCAN with complexity $O(N^2d)$, will scale linearly with the dimensionality of the data set. Combined together, linearity in both the size and dimensionality of the data set makes LMCLUS one of the fastest algorithms in its class, and far superior to ORCLUS and DBSCAN in terms of running time as depicted in Figs. 8.9(b) and 8.10(b).



(a) (b)

Figure 8.9: Scalability, running time vs. data size. (a) LMCLUS's scalability, (b) LMCLUS's scalability relative to ORCLUS and DBSCAN.



(a) (b)

Figure 8.10: Scalability, running time vs. dimensionality of the data. (a) LMCLUS's scalability, (b) LMCLUS's scalability relative to ORCLUS and DBSCAN.

		LMCLUS	ORCLUS
1^{st} data set	mean	0.991	0.921
	median	0.999	0.955
	stdev	0.047	0.1056
2^{nd} data set	mean	0.974	0.993
	median	0.974	0.995
	stdev	0.000053	0.000049

Table 8.3: Stability of LMCLUS compared to ORCLUS in terms of accuracy.

8.8.5 Stability

Being a stochastic algorithm, LMCLUS will produce results that differ with each run depending on the random number generator's seed. For the stability set of tests we used two data sets on which LMCLUS was applied 500 times, and for each run we recorded the accuracy. The first data set, was the sample data of Fig. 7.1. We chose this data set because of the relative proximity of its clusters, which in turn requires the sampled manifolds to be fairly accurate approximations of the true manifolds in which the clusters are embedded. The second data set was a *star* type data set consisting of 3000 points in a 10D space, with 4 clusters embedded in 3D manifolds. The same set of tests were also applied on ORCLUS. The results of this experiment are shown in table 8.3. For each data set we measured the mean, median and standard deviation (stdev) of the algorithm's accuracy. The table shows that LMCLUS is able to maintain high accuracy. Applied on the first data set LMCLUS was able to maintain an average of 0.991 accuracy with only 4 runs which yielded below 0.50 accuracy. Whereas ORCLUS had an average of 0.921 with more than 20 runs below 0.50 accuracy as indicated by the larger standard deviation. Applied on the second data set both algorithms maintained high accuracy, but ORCLUS performed

slightly better, a result consistent with the accuracy experiments in section 8.8.3 where ORCLUS slightly outperformed LMCLUS on the *star* type data sets.

8.9 Experiments with Real Data

In this section we present an evaluation of LMCLUS applied on three real data sets, each coming from a different application domain. As with the experiments conducted on synthetic data, we also applied the other three competing algorithms, ORCLUS, DBSCAN, and HPPC on the three real data sets and compared the results.

Time Series Clustering and Classification

Multivariate time series classification has become increasingly important due to new applications in biomedical signal analysis, DNA microarray analysis, and data mining in temporal databases. In this experiment we applied LMCLUS on a data set obtained from the UCI KDD Archive [82] consisting of 600 points with 60 attributes each, divided into six different classes: *decreasing trend*, *cyclic*, *normal*, *upward shift*, *increasing trend*, and *downward shift*. Fig. 8.11 shows parallel-coordinate plots of ten points from each class. The donors of this data set claim that this is a good data set to test time series clustering because Euclidean distance measures will not be able to achieve good accuracy. The reason is clear, this data set requires an algorithm that is able to identify pattern clusters as discussed in chapter 6. Setting $\Gamma = 0.4$ (the "sensitivity" input parameter) LMCLUS was able to achieve 0.87 with a high of 0.89 accuracy by discovering eight clusters all embedded in 1D linear manifolds, where the cyclic class was divided into three clusters, and where most of the misclassified points came from the decreasing trend class. Table 8.4 shows the corresponding confusion

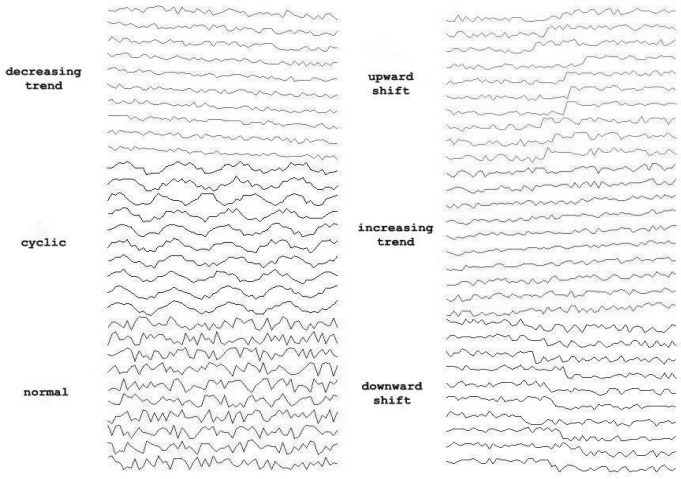


Figure 8.11: Parallel-coordinate plots of a sample of points from a UCI KDD Archive time series data set including six classes of points: decreasing trend, cyclic, normal, upward shift, increasing trend, downward shift.

matrix associated with these results. ORCLUS was only able to achieve 0.50 accuracy, while DBSCAN with extensive tuning of its parameters yielded 0.68 accuracy, and HPPC 0.64 accuracy. These results demonstrate LMCLUS's superiority over the competing algorithms, and its ability to detect pattern clusters that exit in the full space. The reason the clusters were found to be embedded in 1D linear manifolds will be clear in chapter 10.

Handwritten Digit Recognition

The data used in this experiment consisted of preprocessed handwritten digit bitmaps obtained from the UCI Machine Learning Repository [83]. A 32×32 bitmap representing each digit was divided into non-overlapping 4×4 blocks, and the number of "on" pixels in each block was counted. The reduced 8×8 matrix was then converted

output cluster	normal	cyclic	increasing trend	decreasing trend	upward shift	downward shift	total
C_1	0	0	0	**57**	0	0	57
C_2	0	0	**80**	0	1	0	81
C_3	0	0	0	43	0	**99**	142
C_4	0	0	20	0	**98**	0	118
C_5	**99**	0	0	0	0	0	99
C_6	0	**41**	0	0	0	0	41
C_7	0	**23**	0	0	0	0	23
C_8	1	**36**	0	0	1	1	39
total	100	100	100	100	100	100	600

Table 8.4: Confusion matrix obtained by applying LMCLUS on a time series data set.

into a 64-dimensional feature vector whose entries contained integers in the range $[0, 16]$. We then partitioned this data set of 3823 feature vectors into two smaller sets of even $\{0, 2, 4, 6, 8\}$ and odd digits $\{1, 3, 5, 7, 9\}$. Applied on the set of even digits LMCLUS was able to achieve an average of 0.95 accuracy with a high of 0.99, discovering 5 clusters all embedded in one and two dimensional linear manifolds. DBSCAN was able to achieve a high of 0.82 accuracy, while ORCLUS achieved a high 0.85, and HPPC achieved a high of 0.50 accuracy. Applied on the set of odd digits LMCLUS achieved an average of 0.82 accuracy with a high of 0.87, discovering 7 clusters, two clusters for each of the digits 1 and 9, all embedded in one and two dimensional manifolds. DBSCAN achieved a high of 0.58 accuracy, whereas ORCLUS achieved a high of 0.83, and HPPC achieved a high of 0.93 accuracy. We set $\Gamma = 0.4$ for both data sets as input to LMCLUS. We note that the donors of this data set reported they were able to achieve an average of 0.97 classification accuracy by using a supervised K-Nearest Neighbors classification scheme, with a Euclidean distance metric. These results again demonstrate LMCLUS's superiority over the competing

algorithms, and its usefulness in learning structure in the relatively high-dimensional handwritten digit bitmaps data.

E3D Point Cloud Segmentation

Automatic Target recognition (ATR) is a challenging problem made particularly difficult by the ambiguities inherent in attempting to identify 3D objects in 2D data. The E3D (DARPA's "Exploitation of 3D Data") identification system must take as input a 3D point cloud of a military target such as a tank that is produced by a 3D sensor such as the LADAR sensor, and then compares it to a database of highly detailed 3D CAD models. Accomplishing this task generally involves focusing on targets as collections of their constituent parts and reasoning from parts to whole targets. Thus, the first step is to extract the parts, a process generally known as segmentation. In this experiment we demonstrate LMCLUS's capability in identifying these constituent parts. In particular we applied LMCLUS as a segmentation procedure on 3D commercial vehicle point cloud CAD models obtained from ALPHATECH Inc. [84], as these provide a similar level of complexity if not more, to that of military vehicles. In addition the applicability of LMCLUS to this problem results from the fact that the surfaces constituting the vehicles closely correspond to flats which are 2D linear manifolds embedded in a 3D space. We used a slightly modified version of LMCLUS in which the algorithm only searches for clusters embedded in 2D linear manifolds, rather than a range of different dimensionalities. The results of this experiment applied on two different vehicles are depicted in Fig. 8.12 showing images before and after segmentation. For the sake of image clarity the figures shows a 2D view of only a subset of the extracted surfaces. These results clearly demonstrate LMCLUS's ability to identify with high precision 2D linear manifolds.

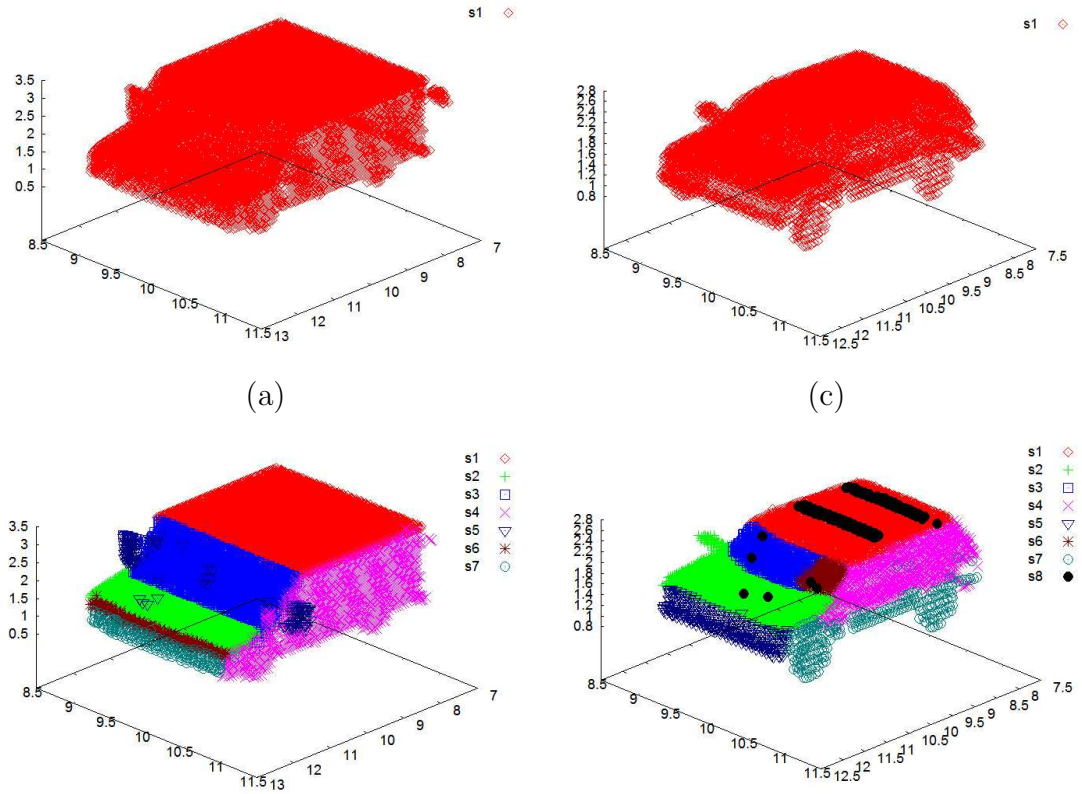


Figure 8.12: (a) 2D view of an Aeromate delivery van (1992) 3D point cloud before segmentation (b) 2D view of the 7 out of 10 surfaces extracted by LMCLUS after segmentation (c) 2D view of a Ford Explorer (1991) 3D point cloud before segmentation (d) 2D view of the 8 out of 12 surfaces extracted by LMCLUS after segmentation.

Chapter 9

Data Description, Density Estimation, and Classification

9.1 Introduction

Data Modeling is the process of extracting useful information out of data and casting is into a model. Data modeling has two main branches, *descriptive modeling* and *predictive modeling*. The aim of *descriptive modeling* is to produce a high-level description of the data, called a *model*, summarizing the data and describing its most important aspects. The aim of predictive modeling is to generate a model with which the value of a variable (feature) whose value was not observed or measured is predicted. When the type of variable is nominal, predictive modeling is called *classification*, and when the variable is numerical, predictive modeling is often called *regression*. The distinction between descriptive and predictive modeling is not always clear, both rely on extracting regularities from the data, and in many cases the model produced by a data modeling process can be used for both descriptive and predictive purposes. As mentioned in section 7.2 one of the ultimate goals of clustering is to generate a description (preferably probabilistic) of the population underlying the data, and to produce a classification method by which a new observation (feature

vector) is labeled. Descriptive models can take on many forms. One of the preferred forms of modeling is that of *probability density estimation*, with which the density of the population in the vicinity of a single object is estimated. The main advantage of describing the data by density estimation is that it facilitates the ability to make statistical inferences about the population underlying the data, and devise probabilistic classification schemes for new observations.

The linear manifold cluster model presented in section 7 is already posed in a probabilistic setting, making the extension to density estimation a lot simpler. The underlying idea is to factorize the density estimation to a weighted sum of individual linear manifold density estimates, i.e., a mixture of linear manifold models. We note that similar approaches have already been discussed, see for example [85, 86, 87, 88, 89, 90, 91, 92]. However most of these methods make strong assumptions that are often violated in real-world problems; such as that the distributions on and off the manifold are Gaussian and that all the manifolds in the mixture have the same dimensionality. In addition, many of these methods rely on an EM scheme that requires the estimation of prohibitively large covariance matrixes.

9.2 Probability Density Estimation

Assuming the clustering yielded K clusters $C_1, C_2, \ldots, C_k$ all following the linear manifold cluster model given in eq. (7.1.1). Then a mixture density for a point $\mathbf{x}$ is defined as

$$p(\mathbf{x}) = \sum_{i=1}^{K} p(\mathbf{x}|C_i)P(C_i), \tag{9.2.1}$$

where $p(\mathbf{x}|C_i)$ is the density of $\mathbf{x}$ assuming it came from cluster C_i, and $P(C_i)$ is the probability that $\mathbf{x}$ arose from cluster C_i. One way to describe this density model is

that each point $\mathbf{x}$ is generated by first randomly picking a linear manifold model or cluster C_i with probability $P(C_i)$ and then drawing the point from the corresponding probability distribution $p(\mathbf{x}|C_i)$.

According to eq. (9.2.1) to estimate the density of a point $\mathbf{x}$ we need separate estimates or density models for each of the components of the mixture, i.e., we need to devise density models for each $p(\mathbf{x}|C_i)$ and $P(C_i)$, where $i = 1, 2, \ldots, K$. $P(C_i)$ is estimated by simply computing the fraction of points coming from cluster C_i, that is,

$$P(C_i) = \frac{|C_i|}{|X|}, \tag{9.2.2}$$

where $|X|$ is the number of points in the data set. According to the linear manifold cluster model of eq. (7.1.1) an estimate of the probability density function or distribution $p(\mathbf{x}|C_i)$ is essentially an estimate of the composite or joint distribution of the random vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\psi}$, representing the distribution of points on the manifold and away from the manifold. However, in order to estimate this joint distribution we first need to estimate the parameters of the linear manifold in which a cluster is embedded. That is, we need to estimate the mean of the linear manifold cluster represented by $\boldsymbol{\mu}$, and the vectors spanning the manifold represented by the matrix B. Estimating the vectors spanning the space orthogonal to the manifold represented by matrix $\overline{B}$ is not necessary given that B is known.

At this point we would like note that one way of the describing the data or the underlying population is by the linear manifolds (their parameters) in which the data is embedded. Although not a probabilistic description, insight to the data is still gained by this type of description, which depending on the final goals may be sufficient enough. Moreover, a classification scheme that simply assigns new points to the closest linear manifold can be devised.

In order to proceed with density estimation certain assumption need to be made. These assumptions create a wide range of possibilities to model the data. At one extreme a parametric approach can be taken similar to that of the linear manifold cluster model. That is, particular functional forms (e.g., Gaussian, uniform) are assumed in advance to model the distributions of points on and off the manifold. Then the problem of modeling the density of the points reduces to the problem of estimating the parameters of the parametric distributions from the data, e.g., estimating $\boldsymbol{\mu}$, Σ, and R. At the other extreme a nonparametric approach where the density estimates are data-driven, in addition to the assumption that all the components involved in the model are conditionally dependent can be taken. In this case more elaborate and computationally more intensive density estimation methods such as *nearest neighbor estimation* [5], *kernel estimation* [5], *graphical models* [93], and others need to be employed. The choice of the method employed depends on a combination of several factors such as the level of complexity of the model desired, the level of accuracy desired, the computational resources available, and on the final goals of the experiment. In the following we discuss a nonparametric and computationally efficient approach that is mid-way between the two extremes and that has experimentally shown to be effective and accurate enough to model the data.

The nonparametric method proposed makes an assumption often referred to as the *naive Bayes assumption*, that the components involved in the model are independent. In other words, the distribution of points on the manifold is independent of the distribution of points off the manifold, and the distribution of points on each of the axis or vectors spanning the manifold is independent of the other. This assumption simplifies the modeling process and allows the modeling of the joint distribution as

a product of marginals. Note that this assumption is consistent with the assumptions made in the linear manifold cluster model, but does not make any parametric assumptions.

As mentioned, to construct a density model we first need to estimate the parameters of each manifold discovered by the clustering algorithm. For each linear manifold cluster C_i the parameter $\boldsymbol{\mu}_i$ is estimated as the sample mean of cluster, i.e.,

$$\boldsymbol{\mu}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}. \tag{9.2.3}$$

The vectors spanning the manifold (columns of matrix B) are obtained through *principle component analysis* (PCA). The aim of PCA can be described in several ways one of which is to find by an orthonormal transformation a new set of variables (components) in decreasing order of importance that explain the maximum amount of variance in the data. According to the linear manifold cluster model, the components having the largest amount of variance are the axis or vectors spanning the manifold, i.e., the column vectors of B, and the components having the least amount of variance are the vectors spanning the space orthogonal to the manifold, i.e. the column vectors of $\overline{B}$. Hence, estimating the column vectors of B is done by finding a set of principle components that explain most of the variance in a linear manifold cluster, a problem which can be formulated as an eigenvalue problem. More formally, assuming d is the dimensionality of the space and k_i is the intrinsic dimensionality of linear manifold cluster C_i. Then the k_i vectors spanning the manifold are the k_i eigenvectors $\mathbf{v}_{i1}, \mathbf{v}_{i2}, \ldots, \mathbf{v}_{ik_i}$ corresponding to the leading (largest) k_i eigenvalues $\lambda_{i1}, \lambda_{i2}, \ldots, \lambda_{ik_i}$ obtained by an eigen-decomposition of the covariance matrix Σ_i of cluster C_i. The

covariance matrix is defined as

$$\Sigma_i = \frac{1}{(|C_i| - 1)} \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \boldsymbol{\mu}_i)(\mathbf{x} - \boldsymbol{\mu}_i)', \tag{9.2.4}$$

and its eigen-decomposition as

$$\Sigma_i = V_i D_i V_i', \tag{9.2.5}$$

where V_i is a $d \times d$ orthonormal eigenvector matrix containing the d eigenvectors $\mathbf{v}_{i1}, \mathbf{v}_{i2}, \ldots, \mathbf{v}_{id}$, and D_i is a $d \times d$ diagonal matrix containing the corresponding d eigenvalues $\lambda_{i1}, \lambda_{i2}, \ldots, \lambda_{id}$ in decreasing order. Since the eigenvalues equal the variances explained by each principle component (eigenvector), selecting the eigenvectors corresponding to the largest eigenvalues will yield the vectors spanning the manifold. However the dimensionality k_i and thus the number of eigenvectors that need to be selected is unknown and must also be estimated. This is typically done by choosing a threshold $\alpha \in [0, 1]$ that specifies the total variance in the data we expect the principle components to explain. More formally

$$k_i = \min \left\{ r \left| \frac{\sum_{j=1}^{r} \lambda_j}{\sum_{j=1}^{d} \lambda_j} \geq \alpha,\ r \in \{1, \ldots, d\} \right. \right\}, \tag{9.2.6}$$

where typical values for α are between 0.8 and 0.9.

Having determined the dimensionality k_i and the parameters $\boldsymbol{\mu}_i$ and $B_i = (\mathbf{v}_{i1}, \mathbf{v}_{i2}, \ldots, \mathbf{v}_{ik_i})$ of each linear manifold, the next step is to model the distributions of points on and away from each manifold. Because of the independence assumption we chose to make, the joint distribution of points on a manifold (their projection) is modeled by the product of the marginal distributions of the projections of points onto each of the k_i principle components spanning the manifold. The projection of a point $\mathbf{x}$ onto the j-th principle component of linear manifold cluster C_i

is given by

$$\mathbf{v}'_{ij}(\mathbf{x} - \boldsymbol{\mu}_i). \tag{9.2.7}$$

To estimate $p_{ij}(\mathbf{v}'_{ij}(\mathbf{x}-\boldsymbol{\mu}_i))$ the true pdf of the projection of points onto the j-principle component of cluster C_i we construct a histogram $h_{ij}(\mathbf{v}'_{ij}(\mathbf{x}-\boldsymbol{\mu}_i))$ of the projections. Hence, the joint distribution of the points (projections) on the manifold is estimated by

$$p_i(B'_i(\mathbf{x} - \boldsymbol{\mu}_i)) = \prod_{j=1}^{k_i} h_{ij}(\mathbf{v}'_{ij}(\mathbf{x} - \boldsymbol{\mu}_i)). \tag{9.2.8}$$

The behavior of points off the manifold are essentially describing the "fuzziness" of the manifolds, which can be modeled in a rather simple way. Instead of modeling the multivariate joint distribution of points off the manifold, the fuzziness of the manifolds can modeled by a univariate distribution of distances of points away from the manifold. The distance of a point $\mathbf{x}$ to linear manifold cluster C_i is given by

$$||(I - B_i B'_i)(\mathbf{x} - \boldsymbol{\mu}_i)||. \tag{9.2.9}$$

To estimate the true pdf of the distances of points to linear manifold cluster C_i, we again construct a histogram $h_i(||(I - B_i B'_i)(\mathbf{x} - \boldsymbol{\mu}_i)||)$ of the distances of points in cluster C_i to the manifold in which they are embedded. Because of the assumption that the distribution of points on and off the manifold are independent the total density estimate of a point $\mathbf{x}$ given that it came from cluster C_i is therefore given by

$$p(\mathbf{x}|C_i) = \left(\prod_{j=1}^{k_i} h_{ij}(\mathbf{v}'_{ij}(\mathbf{x} - \boldsymbol{\mu}_i)) \right) h_i(||(I - B_i B'_i)(\mathbf{x} - \boldsymbol{\mu}_i)||), \tag{9.2.10}$$

and the total mixture density estimate of a point $\mathbf{x}$ is therefore given by

$$p(\mathbf{x}) = \sum_{i=1}^{K} \frac{|C_i|}{|X|} \left(\prod_{j=1}^{k_i} h_{ij}(\mathbf{v}'_{ij}(\mathbf{x} - \boldsymbol{\mu}_i)) \right) h_i(||(I - B_i B'_i)(\mathbf{x} - \boldsymbol{\mu}_i)||). \tag{9.2.11}$$

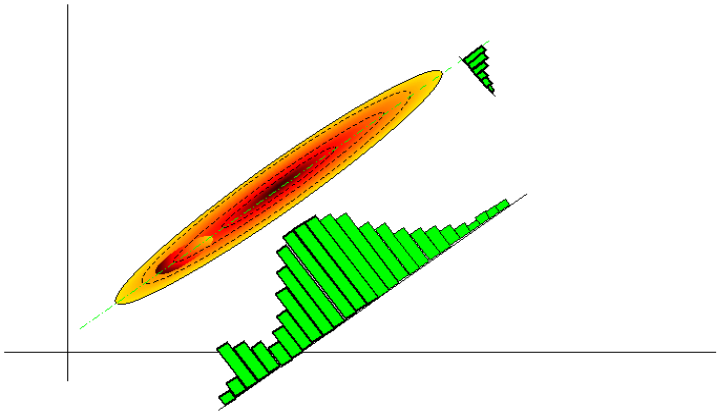


Figure 9.1: Nonparametric linear manifold density estimation by histograms. Darker colors indicate regions of higher density.

An illustration of the concept of linear manifold density estimation by histograms is depicted in Fig. 9.1. The figure shows a one-dimensional linear manifold and the density of points on and away from it. Darker colors indicate regions of higher density. On the manifold the projection of points form an approximate bimodal normal distribution as shown by the corresponding histogram. Off the manifold the distribution of distances follows an approximate chi-squared distribution with a low number of degrees of freedom as illustrated by the smaller histogram.

9.3 Probabilistic Classification

Having devised density and probability models for $p(\mathbf{x}|C_i)$ and $P(C_i)$, whether by the method proposed in section 9.2 or any other alternative method, various probabilistic classification schemes or measures of association can constructed. In the following we

choose to use a Bayesian approach. In the the Bayesian setting $p(\mathbf{x}|C_i)$ is considered to be the *likelihood* of a point $\mathbf{x}$ given class or cluster C_i, and $P(C_i)$ the *prior* of class or cluster C_i. Then using Bayes' rule the *posterior* probability or *Bayesian measure of association* $P(C_i|\mathbf{x}) \in [0,1]$ of a point $\mathbf{x}$ to class or cluster C_i is given by

$$P(C_i|\mathbf{x}) = \frac{p(\mathbf{x}|C_i)P(C_i)}{\sum_{j=1}^{K} p(\mathbf{x}|C_j)P(C_j)}, \tag{9.3.1}$$

and the classification rule $\varphi(\mathbf{x}) \in \{1, 2, \ldots, K\}$ which assigns a class or cluster label to a point can be formulated as

$$\varphi(\mathbf{x}) = \arg\max_i p(C_i|\mathbf{x}). \tag{9.3.2}$$

The classification rule defined in eq. (9.3.2) also establishes the basis for "hard" clustering, i.e., the assignment of an object to one cluster only. However, as discussed in section 4.6, in many applications like gene expression clustering some genes may belong to multiple functional categories, and thus hard clustering may not always be adequate. Unlike hard clustering, "soft" clustering allows for objects to belong to multiple clusters, whereas "fuzzy" clustering associates each object with each cluster by its degree of membership typically ranging in $[0, 1]$. One of the advantages of the Bayesian measure of association defined in eq. (9.3.1) is that it can easily be extended to devise soft and fuzzy assignment schemes. In fact, the fuzzy assignment measure $\varphi(\mathbf{x}) \in [0, 1]$ is exactly equal to the posterior probability, i.e.,

$$\varphi(\mathbf{x}) = p(C_i|\mathbf{x}). \tag{9.3.3}$$

To perform soft clustering a threshold θ must be prespecified, and the soft assignment rule $\varphi(\mathbf{x}) \in \mathcal{P}(\{1, 2, \ldots, K\})$ (power set) can be formulated as

$$\varphi(\mathbf{x}) = \{i | p(C_i|\mathbf{x}) \geq \theta\}, \tag{9.3.4}$$

i.e. a point is assigned to a cluster if its posterior probability is larger than some prespecified threshold.

9.4 Describing Linear Dependencies

An alternative way to describe the data or the underlying population is by the correlations or linear dependencies between the attributes of the data induced by the linear manifold clusters. This is particularly useful when the aim of the clustering is to reveal correlations or linear dependencies among the attributes of the data. In the following we show how such a descriptive model can be derived. The underlying idea is to describe each cluster by a set of linear equations which are easily interpretable and which show linear dependencies between different attributes.

Just as a surface can be defined by the normal vector perpendicular to it. A linear manifold can be defined by the set of vectors which are orthogonal to it. More formally according to the linear manifold model, a linear manifold with parameters μ and B can be defined by

$$\overline{B}'(\mathbf{x} - \boldsymbol{\mu}) = \mathbf{0}. \tag{9.4.1}$$

That is, if a point $\mathbf{x}$ indeed lies on a linear manifold spanned by the column vectors of matrix B, and hence has no component in the space orthogonal to it defined by the column vectors of matrix $\overline{B}$, then its projection to the orthogonal space $\overline{B}'(\mathbf{x} - \boldsymbol{\mu})$ equals zero. The equation system defined in Eq. (9.4.1) exactly describes the linear dependencies in the data induced by a set of points that lie on a linear manifold. To see this more clearly we can rearrange eq. (9.4.1) to give

$$\overline{B}'\mathbf{x} = \overline{B}'\boldsymbol{\mu} \tag{9.4.2}$$

and rewrite the above system of equations as a linear combination of scalars

$$\begin{array}{ccccccccc} b_{k+1,1}x_1 & + & b_{k+1,2}x_2 & +, & \ldots & ,+ & b_{k+1,d}x_d & = & c_{k+1} \\ b_{k+2,1}x_1 & + & b_{k+2,2}x_2 & +, & \ldots & ,+ & b_{k+2,d}x_d & = & c_{k+2} \\ & & & & \vdots & & & & \\ b_{d,1}x_1 & + & b_{d,2}x_2 & +, & \ldots & ,+ & b_{d,d}x_d & = & c_d \end{array} \tag{9.4.3}$$

where $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_k$ are the vectors spanning a k dimensional linear manifold (columns of B), $\mathbf{b}_{k+1}, \mathbf{b}_{k+2}, \ldots, \mathbf{b}_d$ are the vectors spanning the space orthogonal to the manifold (columns of $\overline{B}$), $b_{i,j}$ the j-th component of $\mathbf{b}_i$, $c_{k+i} = \mathbf{b}'_{k+i}\boldsymbol{\mu}$, and x_i the i-th feature or attribute of the data. Note that the number of equations is equal to $d-k$ the dimensionality of the space orthogonal to the linear manifold. Moreover, using *Gauss-Jordan elimination* this set of equations can further be simplified and put in *reduced row echelon* form [94], to produce a unique description of the linear dependencies among the features of the data.

As an example, suppose a $d-1$ dimensional linear manifold was discovered in the data. Then the dependencies in the data can be described by one equation of the form

$$b_{d,1}x_1 + b_{d,2}x_2 +, \ldots, +b_{d,d}x_d = c_d,$$

which if rearranged gives exactly a *regression model* that describes the relationship between a *response* variable (can be any of the $x'_i s$) and a set of *explanatory* or *predictor* variables (the remaining x_i's)

In summary, to find and describe linear dependencies among features of the data the following method can be used.

1. Cluster the data using the linear manifold clustering algorithm presented in chapter 8.

2. For each linear manifold cluster estimate $\boldsymbol{\mu}$ and $\overline{B}$ using the methods discussed in section 9.2.

3. For each linear manifold cluster derive the system of equations $\overline{B}'\mathbf{x} = \overline{B}'\boldsymbol{\mu}$.

4. Using Gauss-Jordan elimination produce the reduced row echelon form of each of the systems of equations.

9.5 Experiments

The data sets used in this experiment consisted of E3D (DARPA's "Exploitation of 3D Data") images of resolution 75mm and 200mm that were converted by a mathematical morphology technique into 120-dimensional feature vectors, where each feature vector corresponds to a pixel in the image. The goal was to build a classifier that would classify pixels in each of the images as either *target* (military vehicles) or *clutter*. For each resolution two data sets were produced: target and clutter, using the class labels which were known a priori. The target sets contained 136066 75mm feature vectors and 21260 200mm feature vectors. The clutter sets contained 229612 75mm and 76748 200mm feature vectors. The classifier was trained using *2-fold cross-validation*. That is, the data sets were randomly partitioned into two sets of equal size, were one set was used for training the classifier and the other for testing its performance. The training and testing sets were then swapped, and the results were averaged.

The classifier was constructed by first clustering each of the data sets using the linear manifold clustering algorithm presented in chapter 8, and then describing the data sets by density estimation models using the modeling technique discussed in section 9.2. The result of this process yielded two mixture density models $p(\mathbf{x}|T)$ and $p(\mathbf{x}|C)$ (corresponding to $p(\mathbf{x})$ of eq. (9.2.11)) representing the likelihood that

a point came from the target and clutter classes respectively. Hence, the underlying assumption was that each of the two classes would better be described using a mixture of models and not a single model. In addition to these two likelihood models the priors $P(T)$ and $P(C)$ were estimated from the size of each of the data sets. Then using Bayes' rule the posterior probabilities of the target and clutter classes were computed by

$$P(T|\mathbf{x}) = \frac{p(\mathbf{x}|T)\ P(T)}{p(\mathbf{x}|T)\ P(T) + p(\mathbf{x}|C)\ P(C)}, \tag{9.5.1}$$

and

$$P(C|\mathbf{x}) = 1 - P(T|\mathbf{x}). \tag{9.5.2}$$

The final classification rule $\varphi(\mathbf{x}) \in \{T, C\}$ is given by

$$\varphi(\mathbf{x}) = \begin{cases} T & \text{if } \frac{P(T|\mathbf{x})}{P(C|\mathbf{x})} \geq \theta \\ C & \text{otherwise,} \end{cases} \tag{9.5.3}$$

where θ is *discrimination threshold* with which a trade off between selectivity and sensitivity (true positives and false positives) can be obtained. The value for θ that optimizes a certain criteria such as minimizing the overall classification error rate is obtained through *Receiver Operating Characteristics* (ROC) analysis [5]. ROC analysis is a method originating from *signal detection theory* that examines the performance of a classifier as a function of the *false-alarm* and *misdetect* rates by varying a discrimination threshold. The results of the analysis are typically illustrated by an *ROC curve*, a plot of the false-alarm rates versus the misdetect rates for varying values of θ, where a smaller area under the curve generally indicates a better classifier. Having no particular criteria defined in advance the threshold is selected as the point that is closest to the origin in the ROC curve, i.e. a threshold that minimizes the

false-alarm and misdetect rates simultaneously without giving preference to any one of them.

In our case we construct the ROC curve as follows: for each feature vector $\mathbf{x}_i$ in the testing set we compute the ratio $\theta_i = \frac{P(T|\mathbf{x}_i)}{P(C|\mathbf{x}_i)}$ and using these ratios we construct the set of discrimination thresholds $\Theta = \{\theta_1, \theta_2, \ldots, \theta_N\}$. Then we apply the classification rule $\varphi(\mathbf{x})$ on each of the thresholds in Θ and compute the following contingency table for each of the thresholds,

		assigned	
		target	clutter
true	target	w	x
	clutter	y	z

were w is the number of target points that were correctly assigned target by $\varphi(\mathbf{x})$ using θ_i, x is the number of target points that were incorrectly assigned clutter, y is the number of clutter points that were incorrectly assigned target, and z is the number of clutter points that were correctly assigned clutter. Using this table we then compute for each θ_i the probability of false-alarm defined as

$$P(\text{false-alarm}) = \frac{y}{y+z}, \tag{9.5.4}$$

and the probability of misdetect defined as

$$P(\text{misdetect}) = \frac{x}{w+x}, \tag{9.5.5}$$

and plot the corresponding probabilities in the ROC plot.

The final results of our experiment are shown by the ROC curves of Figs. 9.2 and 9.3. The figures show a generally good performance of the classifier, measured by the area under the curve, where at the points closest to the origin the average overall

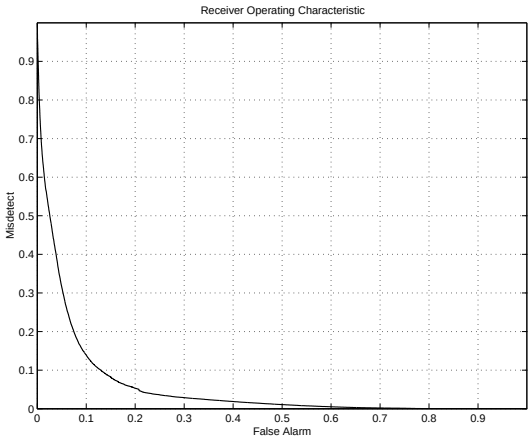


Figure 9.2: ROC curve for the 75mm E3D data sets.

classification error defined as

$$\text{classification error} = \frac{x+y}{w+x+y+z} \tag{9.5.6}$$

is very close to 0.1 for 75mm data sets and 0.17 for the 200mm data sets.

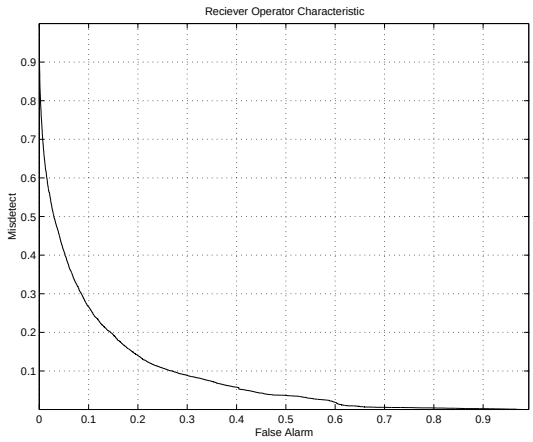


Figure 9.3: ROC curve for the 200mm E3D data sets.

Chapter 10

Pattern and Correlation Clusters as Instances of Linear Manifolds

10.1 Introduction

As discussed in chapter 6, unlike the classical or the subspace clustering paradigms that seek to identify groups of points with similar values in either the full space or subspaces of the data, the pattern or correlation clustering paradigm seeks to identify groups of objects that exhibit coherent or correlated behavior patterns in subspaces of the data. Interest in this type of clustering has emerged due to new areas of application such as gene expression microarray analysis, collaborative filtering, and web mining. In gene expression clustering the goal is to identify groups of genes that exhibit similar expression patterns under some subset of conditions (axis-parallel subspaces of the data), from which gene function or regulatory mechanisms may be inferred. In collaborative filtering systems, sets of customers or users with similar interest patterns need to be identified so that future interests can be predicted and recommendations be made. From a correlation point of view, groups of objects exhibiting coherent behavior patterns (pattern clusters) induce certain types linear correlations among subsets of features, so the identification of correlations is a means

by which pattern clusters can be discovered. Moreover, the identification of correlations can also be used for local dimensionality reduction by eliminating correlated features, and to learn linear dependencies among features of the data.

The most widely studied behavior patterns in the context of pattern clustering are the *shift* and *scaling* patterns, which both induce only positive correlations. In the case of a shift pattern (also called *slope one* pattern) the behavior pattern of one object under a subset of features is offset from another by some constant, whereas in the case of the scaling pattern the behavior pattern of one object is a scaler multiple of another. As mentioned in section 6.9, it has been suggested other types of information carrying patterns such as patterns inducing negative correlations are completely overlooked by most clustering methods. Another draw back with current state of the art algorithms is that they are not flexible enough to identify different types of patterns simultaneously. They assume a unique underlying pattern or cluster model, while overlooking or rejecting the possibility that other types of information carrying patterns may co-exit in the data. This assumption typically leads to a large bias in the clustering results, and therefore a method that is able to take into account and identify all types of patterns or equivalently identify large correlations both positive and negative will be beneficial.

In section 7.2 the motivations for the linear manifold clustering paradigm were presented. One of the main motivations is that the linear manifold cluster model is a generalization of many other more specific cluster models. In other words, other cluster models such as classical clusters, subspace clusters, and pattern or correlation clusters are all instances of the linear manifold cluster model. Therefore in the case of pattern or correlation clustering, by searching for linear manifolds rather than specific

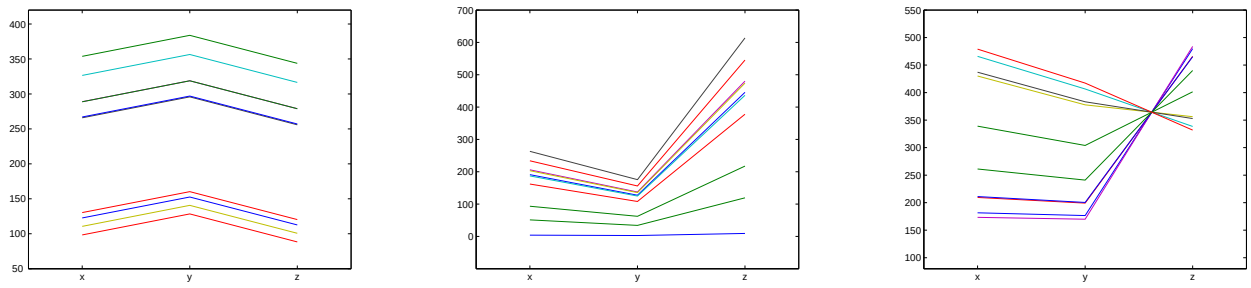
types of patterns or correlations it is possible to rectify the problems associated with current state of the art methods aiming at pattern clustering. That is, being able to identify all types of patterns and correlations concurrently without being restricted to one specific type.

The proof of classical and subspace clusters as instances of linear manifolds is a simple manipulation of the linear manifold cluster model and was already presented in section 7.2. The proof of pattern clusters as instances of linear manifolds is more elaborate and will be presented in this chapter. Pattern clusters give rise to special cases of correlation and linear dependencies. In this chapter we discuss two additional results pertaining to correlations and linear dependencies, that extend the applicability of linear manifolds beyond the special case of pattern clusters. One result showing that any type of linear dependency in the data, and not only those generated by pattern clusters, can be captured by linear manifold clusters. The other showing that a set of points that induce any type of large correlations among a set of features can be captured by one-dimensional linear manifolds. The last results will also be used as a basis for a new clustering algorithm presented in chapter 11, which is a derivative of the linear manifold clustering algorithm, but which is aimed specifically at pattern and correlation clustering.

10.2 Pattern Clusters

The main idea of pattern clusters as instances of linear manifolds, is that within the set of relevant features (subspaces) pattern clusters manifest themselves geometrically as lines which are one-dimensional linear manifolds. The difference between the different types of pattern clusters is the orientation of the line formed by the cluster and its

translation from the origin. For example, shift clusters are oriented in the direction of the vector **1**, i.e., have slope one, hence the name slope-one clusters given to them in the collaborative filtering literature. The line formed by a scaling pattern cluster on the other hand may be oriented in an arbitrary positive direction but must pass through the origin. Patterns inducing negative correlations are manifested by lines which have a negative direction, i.e., a negative slope. In the full space these lines, representing different types of pattern clusters, transform into linear manifolds of higher dimension which are parallel to some of the coordinate axes. More specifically, if d is the dimensionality of the data space, and k the number of relevant features or the dimensionality of the subspace in which a pattern cluster exists, then the dimensionality of the linear manifold formed by the pattern cluster in the full space will be $d-k+1$, where $d-k$ of its spanning basis vectors are parallel to some set of $d-k$ coordinate axes. Figs. 10.1-10.3 illustrate these ideas. Fig. 10.1 shows parallel coordinate plots of three different (shift, scaling, and negative correlation inducing) pattern clusters embedded in a three-dimensional space, while Fig. 10.2 shows the geometry of the same three pattern clusters, in the corresponding three-dimensional data space ($d = k = 3$). Notice that all three pattern clusters share the geometry of lines, which are differentiated by their orientation and translation from the origin. The figure also shows a regular cluster to emphasize the geometrical difference between the shapes formed by pattern and regular clusters. Fig. 10.3 exemplifies the concept of subspace pattern clusters appearing as higher dimensional linear manifolds in the full space, where $d = 3$ and $k = 2$. Fig. 10.3(a) shows a parallel coordinate plot of a shift pattern cluster embedded in a two-dimensional subspace (spanned by the x and y axis) of a three-dimensional space. In the subspace of relevant dimensions the



(a) shift pattern (b) scaling pattern (c) negative correlation inducing pattern

Figure 10.1: Parallel coordinate plots of three different pattern clusters.

points exhibit coherent behavior, whereas in the third dimension the points behave haphazardly. Fig. 10.3(b) on the other hand, shows the same set of point in the three-dimensional data space. Notice that in the subspace of relevant dimensions the points still form a line, but in the full space a two-dimensional linear manifold which is parallel to the z coordinate axis.

To prove that pattern clusters are instances of the linear manifold cluster model we first need to restate the pattern cluster models (described by scalars) as vector models. Relying and extending the notation used to define the linear manifold cluster model, and the pattern cluster models specified in eq. (6.6.1) and eq. (6.6.5), we denote by X a set of d-dimensional points that form a pattern cluster in some k-dimensional subspace, $\mathbf{x}$ a $d \times 1$ vector representing some point in X, and B and $\overline{B}$ two matrices whose columns span k and $d-k$ dimensional subspaces respectively, which are orthogonal complements of each other.

Definition 10.2.1 (Vectorized Shift Pattern Model)**.** *Each* $\mathbf{x} \in X$*, a shift pattern cluster can be modeled by,*

$$\mathbf{x} = B(\mathbf{1}_k\mu + \mathbf{1}_k\alpha + \boldsymbol{\beta} + \boldsymbol{\epsilon}) + \overline{B}(\overline{\boldsymbol{\mu}} + \boldsymbol{\lambda}). \tag{10.2.1}$$

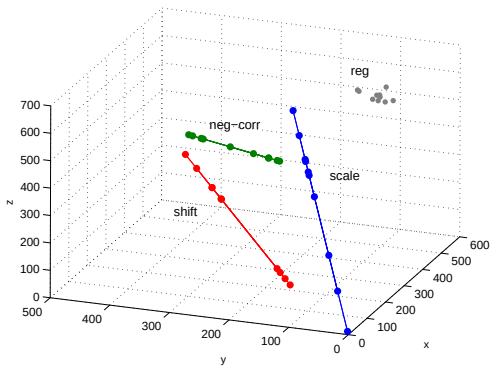
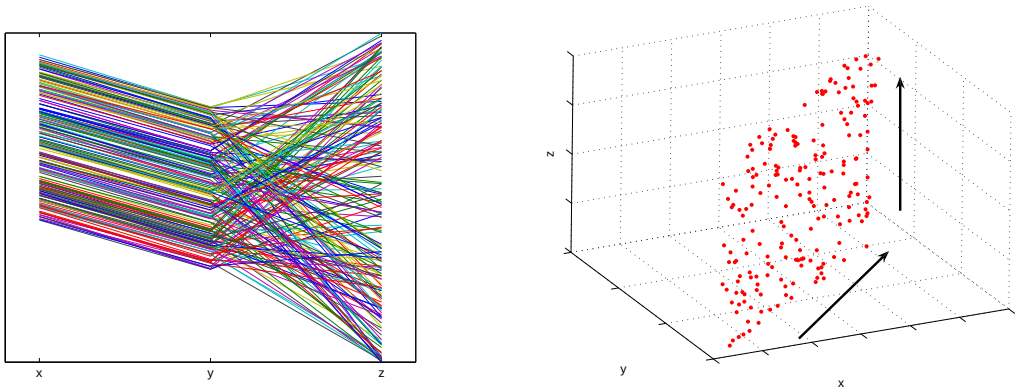


Figure 10.2: The geometry of the three pattern clusters from Fig. 10.1 in the data space. All are manifested by lines but are oriented and translated differently depending on the type of pattern they manifest. A regular cluster (reg) is also plotted to emphasize the difference between pattern and regular clusters.



(a) parallel coordinate plot

(b) data space plot

Figure 10.3: A shift pattern embedded in a 2D subspace. In the full space the pattern is manifested by a 2D linear manifold.

Explanation: In the k-dimensional subspace in which the cluster is embedded each point (a $k \times 1$ vector) can be modeled in vector form by

$$\mathbf{1}_k\mu + \mathbf{1}_k\alpha + \boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

In this case α is considered a random scaler with mean zero representing the residual effect of some level of factor α. $\boldsymbol{\beta} = (\beta_1, \ldots \beta_k)'$ a constant vector containing the residual effects of the k levels of factor β, and $\boldsymbol{\epsilon} = (\epsilon_1, \ldots \epsilon_k)'$ a random vector containing the errors associated with the corresponding level (point). Since it is common to assume that the errors a normally distributed with mean zero and some variance σ^2, $\epsilon \sim N(\mathbf{0}, \sigma^2 I_k)$. Therefore the template or mean of the cluster in the subspace is equal to $\mathbf{1}_k\mu+\boldsymbol{\beta}$, in which case α can be considered the shift or translation of a point from the template. Since the pattern clusters and the subspaces in which they are embedded are defined in terms of the original measurement features, the matrices B and $\overline{B}$ will contain subsets of columns of a d-dimensional identity matrix, where each column corresponds to some measurement feature or coordinate axis. Hence, by prefixing the first component of the above model by B we obtain a point's representation in the corresponding subspace in which the cluster is embedded. In the complementary subspace the points are assumed to behave haphazardly. This is modeled by the second component of the model

$$\overline{B}(\overline{\boldsymbol{\mu}} + \boldsymbol{\lambda}),$$

where $\overline{\boldsymbol{\mu}}$ is a $d - k \times 1$ vector representing the mean of the points in this subspace, and $\boldsymbol{\lambda}$ a $d - k \times 1$ random vector whose entries are each uniformly distributed with mean zero and large variance. $\boldsymbol{\lambda}$ essentially represents the random translation of the points from the mean $\overline{\boldsymbol{\mu}}$. Using the vectorized shift pattern model of eq. (10.2.1) we

can now prove that a shift pattern cluster is an instance or special case of a linear manifold cluster.

Proposition 10.2.1. *Every point* $\mathbf{x}$ *in a d-dimensional space that fits a shift pattern cluster as modeled by eq. (10.2.1), where the cluster is embedded in some k-dimensional subspace, also fits a linear manifold cluster, where the dimensionality of the linear manifold is* $d - k + 1$*, and the linear manifold cluster model is given by,*

$$\mathbf{x} = (B|\overline{B}) \begin{pmatrix} \mathbf{1}_k\mu + \boldsymbol{\beta} \\ \overline{\boldsymbol{\mu}} \end{pmatrix} + \left(B\frac{\mathbf{1}_k}{\sqrt{k}}\,|\overline{B}\right) \begin{pmatrix} \sqrt{k}\alpha + \frac{\mathbf{1}_k'}{\sqrt{k}}\boldsymbol{\epsilon} \\ \boldsymbol{\lambda} \end{pmatrix} + B\left(I_k - \frac{\mathbf{1}_k\mathbf{1}_k'}{k}\right)\boldsymbol{\epsilon}. \tag{10.2.2}$$

Proof: multiplying out the three terms in eq. (10.2.2) gives eq. (10.2.1) showing that the two models are equivalent. □

The three components appearing the model of eq. (10.2.2) correspond to the three components of the general linear manifold cluster model specified in eq. (7.1.1). That is, they correspond to the mean or translation of the linear manifold, the modeling of points on the manifold, and the modeling points off the manifold respectively. However, in this particular instance the manifold is spanned by a specific set of vectors. By inspecting the matrix prefixing the second component, we see that within the set of relevant features or relevant subspace in which the pattern cluster is embedded the points form a one-dimensional linear manifold (line) spanned by the unit norm vector $\frac{\mathbf{1}_k}{\sqrt{k}}$, i.e., a vector with slope or direction one. The second component also reveals that in the full space the manifold is spanned by the orthonormal vectors of the matrix $\left(B\frac{\mathbf{1}_k}{\sqrt{k}}\,|\overline{B}\right)$. The dimensionality of the linear manifold is equal to the rank of this matrix which equals $1 + d - k$. Since part of the manifold is spanned by the columns of the matrix $\overline{B}$, which consists of columns of the identity matrix, the manifold will be parallel to a set of $d-k$ measurement axes. The third component includes the vectors spanning the space orthogonal to the manifold. Noteworthy is the fact that the error

term of eq. (10.2.1) is absorbed both by the manifold and the space orthogonal to it.

Along the same lines, scaling patterns can also be shown to be instances of linear manifolds.

Definition 10.2.2 (Vectorized Scaling Pattern Model)**.** *Each* $\mathbf{x} \in X$*, a scaling pattern cluster can be modeled by,*

$$\mathbf{x} = B(\mu\boldsymbol{\beta}\alpha + \boldsymbol{\epsilon}) + \overline{B}(\overline{\boldsymbol{\mu}} + \boldsymbol{\lambda}). \tag{10.2.3}$$

Explanation: All the terms appearing in this model are defined in the same way as in the vectorized shift pattern model of eq. (10.2.1). In this case in the k-dimensional subspace embedding of the cluster each point can be modeled by

$$\mu\boldsymbol{\beta}\alpha + \boldsymbol{\epsilon}.$$

Note that now the effects of the two factors α and β are multiplicative rather than additive, but the effect of the error ϵ remains additive, which as mentioned earlier is a reasonable assumption. Also note that in this case the template (mean) of the cluster in the subspace is equal to $\mu\boldsymbol{\beta}$, and α can be considered the multiplicative term which scales the template to produce another point. The matrices B and $\overline{B}$ are as before used to give a full space representation of a point as a sum of its components in the subspaces of relevant and irrelevant features respectively, where in the subspace of irrelevant features the points are again assumed to behave randomly. Using this model we can now also prove that a scaling pattern cluster is an instance of a linear manifold cluster.

Proposition 10.2.2. *Every point* $\mathbf{x}$ *in a* d*-dimensional space that fits a scaling pattern cluster as modeled by eq. (10.2.3), where the cluster is embedded in some* k*-dimensional subspace, also fits a linear manifold cluster, where the dimensionality of the linear manifold is* $d - k + 1$*, and the linear manifold cluster model is given by,*

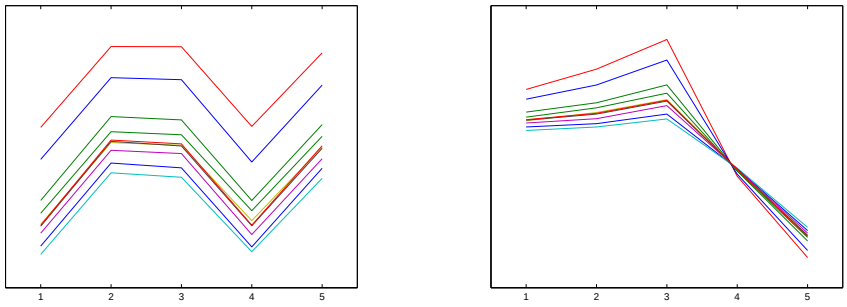
$$\mathbf{x} = \overline{B}\overline{\boldsymbol{\mu}} + \left(B\frac{\mu\boldsymbol{\beta}}{||\mu\boldsymbol{\beta}||}\,|\overline{B}\right)\begin{pmatrix} ||\mu\boldsymbol{\beta}||\alpha + \frac{\mu\boldsymbol{\beta}'}{||\mu\boldsymbol{\beta}||}\boldsymbol{\epsilon} \\ \boldsymbol{\lambda} \end{pmatrix} + B\left(I_k - \frac{(\mu\boldsymbol{\beta})(\mu\boldsymbol{\beta})'}{||\mu\boldsymbol{\beta}||^2}\right)\boldsymbol{\epsilon}. \tag{10.2.4}$$

Proof: Same as with the previous proof, multiplying out the three terms in eq. (10.2.4) gives eq. (10.2.3) showing that the two models are equivalent. □

Notice that in the case of the scaling pattern as opposed to the shift pattern, in the subspace of relevant features defined by the column vectors of B the points still form a one-dimensional linear manifold, but one which is spanned by the unit norm vector $\frac{\mu\beta'}{||\mu\beta||}$. Thus the geometrical difference between the two patterns in the subspace of relevant features is that in the case of the shift pattern the orientation of the manifold is in the direction of the vector $\mathbf{1}$ and the manifold does not necessarily have to pass through the origin, whereas in the case of the scaling pattern the the manifold can be orientated in any direction which is defined by the vector $\mu\boldsymbol{\beta}$, but the manifold must pass through the origin as α can equal zero. Another difference is that in the case of the scaling pattern the translation of the manifold cluster (first component in the model) depends only on the mean of the points in the subspace spanned by the irrelevant features. In the full space the scaling pattern just as the shift pattern forms a $d-k+1$-dimensional linear manifold, where $d-k$ of it spanning vectors are parallel to a subset of $d-k$ coordinate axes. In addition the error term $\boldsymbol{\epsilon}$ as with the shift pattern is also absorbed by the manifold and its complementary subspace.

As mentioned, pattern clusters are defined in terms of the original measurement features, which is expressed in the models by the fact that the matrices B and $\overline{B}$ contain columns of the identity matrix, each defining a different coordinate axis or measurement feature. The main reason for this requirement is that the resulting pattern clusters are readily interpretable, e.g., when the expression level of one gene rises under a specific condition so will the expressions of other genes in the same cluster.

However, it is possible that complex systems such as the system of gene activity are a result of more complex processes, where for example the expression levels under a certain experimental condition are a result of a linear combination of other expression levels, an approach that is very popular in text mining and information retrieval systems [95]. In this case "pseudo" variables which are a linear combination of the original variables (words or documents) are used to infer new "concepts" (meanings) that bring out the "latent" semantic structure of a vocabulary used in a corpus, which would be obscured by word variability otherwise. In our case, to allow for this possibility, i.e., having new features that are a linear combination of the original measurement features, all that is necessary is to replace the columns of the matrices B and $\overline{B}$ in the pattern cluster models by any set of orthonormal vectors that span $\mathbb{R}^d$. Note that by changing these matrices the geometry of the resulting pattern clusters does not change; they are still characterized by linear manifolds. Hence, linear manifolds are able to capture more complex behavior patterns. Fig. 10.4 for example shows what a shift pattern that is only visible in some linear combination of the original measurement features and embedded in a one-dimensional linear manifold (Fig. 10.4(a)), would look like when viewed through the original measurement features (Fig. 10.4(b)). The figure reveals that in the original feature space the features are still highly correlated, but rather than being only positively correlated they may also be negatively correlated (features 1, 2, and 3 are negatively correlated with 4 and 5).



(a) new feature space (b) original feature space

Figure 10.4: A Shift pattern induced by a linear combination of the original measurement features.

10.3 Correlations and Linear Dependencies

Pattern clusters (shift and scaling) give rise to specific instantiations of correlations and linear dependencies. From a correlation point of view, as mentioned already, perfect pattern clusters induce perfect positive correlation between each pair of features that underly the pattern in the space of relevant features, i.e., for each pair of features $x_i,\ x_j$ where $i, j \in \{1, 2, \ldots, k\}$, $\rho_{ij} = 1$. From the point of view of linear dependencies, the patterns can be described by a set of $\begin{pmatrix} k \\ 2 \end{pmatrix}$ linear equations, each describing the linear dependency between a pair of different features, where for the case of the shift pattern the dependency is of the form $x_j = x_i + c_{ij}$ (derived from eq. (10.2.1)), and for the case of the scaling pattern of the form $x_j = b_{ij}x_j$ (derived from eq. (10.2.3)), where c_{ij} and b_{ij} are constant coefficients. However, correlations can also be negative, and linear dependencies can be more complex and elaborate than the ones discussed above. For example, the linear dependency can be in the form of one or more features depending on a combination of several other features,

it can be in the form of a system of equations, or it can be in the simple yet most popular form of a regression model. The fact that linear manifolds give rise to linear dependencies (a set of linear equations) was discussed in section 9.4. The converse can be shown to be true by a reverse process, but can only be materialized when the a specific set linear equations is specified. In this section we illustrate this idea by a specific example of linear dependencies in the form of a regression model. We show that the linear dependency formed by a regression equation gives rise to a specific linear manifold. We also show that from a correlation point of view, large correlations whether positive or negative between a set of features give rise to one-dimensional linear manifolds (lines). This result is then used as a basis for an alternative algorithm described in chapter 11, which is a derivative of the linear manifold clustering algorithm aimed specifically at identifying large correlations and pattern clusters, and which rectifies the shortcomings of the linear manifold clustering algorithm when applied to the problem of correlations and pattern clustering. These shortcomings will be discussed in section 10.4.3. It should also be strongly noted that linear dependencies do not imply large correlation in its strict and formal sense. Only when the linear dependency involves two features and is of the form $x_j = b_{ij}x_i + c_{ij}$ (of which the shift and scaling dependencies are a special case) will there be large correlation. When more features are involved correlations may exist but may not necessarily be large. For example, in the regression model the correlation between the dependent and an independent variable is not necessarily large.

According to the regression model a response (dependent) variable x_d is explained by a set of explanatory (predictor or independent) variables $x_1, x_2, \ldots, x_{d-1}$ in the

following form

$$x_d = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_{d-1} x_{d-1}, \tag{10.3.1}$$

where $\beta_0, \beta_1, \ldots, \beta_{d-1}$ are called the coefficients of the model.

Proposition 10.3.1. *Every point* $\mathbf{x}$ *in a d-dimensional space that fits a regression model as defined by eq. (10.3.1), also fits the linear manifold cluster model constructed from the coefficients of the regression model, where the dimensionality of the linear manifold is* $d-1$.

Proof: Most regression assumptions are concerned with the residuals or error terms. When these are upheld making additional assumptions about the independent variables such them being fixed or random is not problematic. So, to account for the fact that the linear manifold cluster model is a stochastic model, we assume that the independent variables are randomly distributed, so that their projection onto the manifold can be described by the $d-1 \times 1$ random vector $\boldsymbol{\lambda}$. From section 9.4 we know that because the points are described by one equation the dimensionality of the linear manifold which the points fit is $d-1$ and that the dimensionality of the space orthogonal to the manifold is 1. We also know from section 9.4 that an alternative way to describe the manifold is by its normal vector $\overline{B}$ which must satisfy the equation

$$\overline{B}'\mathbf{x} = \overline{B}'\boldsymbol{\mu}.$$

Eq. (10.3.1) can be rearranged as

$$\beta_1 x_1 + \beta_2 x_2 + \ldots \beta_{d-1} x_{d-1} - x_d = \beta_0,$$

so that the left hand side and the right and side will equal $\overline{B}'\mathbf{x}$ and $\overline{B}'\boldsymbol{\mu}$ respectively.

Since

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d = \beta_0 + \sum_{i=1}^{d-1} \beta_i x_i \end{pmatrix},$$

by letting

$$\overline{B} = \frac{1}{\sqrt{1 + \beta_1^2 +, \ldots, +\beta_{d-1}^2}} \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_{d-1} \\ -1 \end{pmatrix},$$

and

$$\boldsymbol{\mu} = \frac{-\beta_0}{\sqrt{1 + \beta_1^2 +, \ldots, +\beta_{d-1}^2}} \overline{B}.$$

we get

$$\overline{B}'\mathbf{x} = \overline{B}'\boldsymbol{\mu} = \frac{-\beta_0}{\sqrt{1 + \beta_1^2 +, \ldots, +\beta_{d-1}^2}}$$

satisfying the equation that defines a linear manifold. Having found $\overline{B}$ the vector spanning the space orthogonal to the manifold, we can now derive the remaining $d-1$ vectors which span the manifold (not unique) and form the matrix B. Using $\boldsymbol{\mu}$, B, $\overline{B}$ we can now define the linear manifold cluster model which the points fit as

$$\mathbf{x} = \frac{-\beta_0}{\sqrt{1 + \beta_1^2 +, \ldots, +\beta_{d-1}^2}} \overline{B} + B\boldsymbol{\lambda} + \overline{B}\boldsymbol{\psi}, \tag{10.3.2}$$

where $\overline{B}\boldsymbol{\psi}$ models the fuzziness of the manifold introduced by adding an error or residual term ϵ to the regression model. □

We now establish the connection between correlation and one-dimensional linear manifolds (lines). We would like to show that if a set of points induce large correlations between a set of features, then they must also fit the linear manifold cluster model,

where the dimension of the linear manifold is one. Recall that pattern clusters induce large correlations. However, these correlations are very specific, i.e., only positive and resulted by very specific linear dependencies, which are manifested in the data space by specific lines, e.g., slope one lines. The following result, which will be used as a basis for the algorithm presented in chapter 11 allows us to capture any type of correlation formed by the linear dependency $x_j = b_{ij}x_i + c_{ij}$, by searching for lines of arbitrary orientation and translation.

Proposition 10.3.2. *A set of points induce perfect correlations among a set of k features if and only if the set of points perfectly fit a one-dimensional linear manifold (line) in this set of features.*

Proof: Following the linear manifold cluster model specified in eq. (7.1.1) a set of points will perfectly fit a 1D linear manifold if they do not include an error term which translates them away from the manifold. Formally, a perfect fit implies that each point within the set of k features can be modeled by

$$\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\beta}\phi,$$

where $\boldsymbol{\beta}$ is a $k \times 1$ vector (replacing the matrix B), which spans the manifold, and ϕ a random scalar (replacing the random vector $\boldsymbol{\lambda}$) with mean zero representing the translation (on the manifold) of the point from the mean. By perfect correlations we mean that the correlation coefficient ρ_{ij} between any pair of features i and j or equivalently the random components x_i and x_j of the random vector $\mathbf{x}$, are equal to 1 or -1. The proposition can now be stated formally as

$$\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\beta}\phi \Leftrightarrow \forall i, j \in \{1, \ldots, k\}\ \rho_{ij} = \pm 1.$$

($\Rightarrow$): Given $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\beta}\phi$, each component or feature of $\mathbf{x}$ denoted by x_i is equal

to $\mu_i + \phi\beta_i$ where μ_i and β_i are the i-th components of vectors $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$. Therefore,

$$\mathrm{Var}[x_i] = \mathrm{Var}[\mu_i + \phi\beta_i] = \beta_i^2 \mathrm{Var}[\phi],$$

$$\begin{aligned}
\mathrm{Cov}(x_i, x_j) &= \mathrm{E}[x_i x_j] - \mathrm{E}[x_i]\mathrm{E}[x_j] \\
&= \mathrm{E}[(\mu_i + \phi\beta_i)(\mu_j + \phi\beta_j)] - \mu_i\mu_j \\
&= \mathrm{E}[\mu_i\mu_j + \mu_i\phi\beta_j + \mu_j\phi\beta_i + \phi^2\beta_i\beta_j] - \mu_i\mu_j \\
&= \mu_i\mu_j + 0 + 0 + \beta_i\beta_j E[\phi^2] - \mu_i\mu_j \\
&= \beta_i\beta_j \mathrm{E}[\phi^2] \\
&= \beta_i\beta_j \mathrm{Var}[\phi],
\end{aligned}$$

and therefore

$$\begin{aligned}
\rho_{ij} &= \frac{\mathrm{Cov}(x_i, x_j)}{\sqrt{\mathrm{Var}[x_i]}\sqrt{\mathrm{Var}[x_j]}} \\
&= \frac{\beta_i\beta_j \mathrm{Var}[\phi]}{\sqrt{\beta_i^2 \mathrm{Var}[\phi]}\sqrt{\beta_j^2 \mathrm{Var}[\phi]}} \\
&= \frac{\beta_i\beta_j \mathrm{Var}[\phi]}{\sqrt{\beta_i^2\beta_j^2}\mathrm{Var}[\phi]} \\
&= \frac{\beta_i\beta_j}{|\beta_i\beta_j|} = \pm 1.
\end{aligned}$$

($\Leftarrow$): Given $\forall i, j \in \{1, \ldots, k\}$ $\rho_{ij} = \pm 1$. Assume $\mathrm{Var}[x_i] = \alpha_i^2$ and $\mathrm{Var}[x_j] = \alpha_j^2$, and let $\frac{x_i}{\alpha_i} - \frac{x_j}{\alpha_j}$ be a new random variable (r.v.). Then,

$$\begin{aligned}
\mathrm{Var}\left[\frac{x_i}{\alpha_i} - \frac{x_j}{\alpha_j}\right] &= \mathrm{Var}\left[\frac{x_i}{\alpha_i}\right] + \mathrm{Var}\left[\frac{x_j}{\alpha_j}\right] - 2\mathrm{Cov}\left(\frac{x_i}{\alpha_i}, \frac{x_j}{\alpha_j}\right) \\
&= \frac{\mathrm{Var}[x_i]}{\alpha_i^2} + \frac{\mathrm{Var}[x_j]}{\alpha_j^2} - 2\frac{\mathrm{Cov}(x_i, x_j)}{\alpha_i\alpha_j} \\
&= 1 + 1 - 2\rho_{ij} \\
&= 2(1 - \rho_{ij}).
\end{aligned}$$

Now $\rho_{ij} = 1$ implies that $\text{Var}\left[\frac{x_i}{\alpha_i} - \frac{x_j}{\alpha_j}\right] = 0$ which in turn implies that the r.v. $\frac{x_i}{\alpha_i} - \frac{x_j}{\alpha_j} = c$, i.e., equals a constant, from which we can then establish a linear relationship between x_i and x_j of the form $x_j = b_{ij}x_i + c_{ij}$. Similarly, $\rho_{ij} = -1$ implies that $\text{Var}\left[\frac{x_i}{\alpha_i} + \frac{x_j}{\alpha_j}\right] = 0$, which again establishes a linear relationship between x_i and x_j of the form $x_j = b_{ij}x_i + c_{ij}$. By introducing $\beta_i, \beta_j, \mu_i, \mu_j$, the r.v. ϕ, and letting $b_{ij} = \beta_j/\beta_i$ and $c_{ij} = -\beta_j\mu_i/\beta_i + \mu_j$, and then substituting them into $x_j = bx_i + c$ we get

$$\frac{x_i - \mu_i}{\beta_i} = \frac{x_j - \mu_j}{\beta_j}.$$

Since both sides of the equation define a r.v. we may choose to call this r.v. ϕ, yielding that

$$x_i = \mu_i + \beta_i\phi \quad \text{and} \quad x_j = \mu_j + \beta_j\phi.$$

Now collecting all the scalar equations of the form above, and putting then in vector format gives the final result that $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\beta}\phi$. □

Using elements of the this proof it can also be shown that the more a set of points deviates from a predefined 1D linear manifold (the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$ are fixed) of the form $\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\beta}\phi + \overline{\boldsymbol{\beta}}\boldsymbol{\epsilon}$, where $\overline{\boldsymbol{\beta}}\boldsymbol{\epsilon}$ models the deviation of the points from the line, the less correlated the features underlying the points will be, where the amount of correlation is dependent on the covariance of $\boldsymbol{\epsilon}$.

10.4 Experiments

As discussed in section 10.2 pattern clusters manifest themselves in the data space as linear manifolds, making the linear manifold clustering paradigm not only applicable to the problem of pattern clustering but also advantageous. In this section we present a set of experiments on real data sets used to demonstrate the effectiveness and

applicability of the linear manifold clustering paradigm to the problem of pattern clustering.

Because the majority of pattern clustering methods are aimed at gene expression analysis, and in order to compare our results with other known results, we conducted extensive tests on two typical data sets that have become standard benchmarking data sets for clustering gene expression data: the *yeast Saccharomyces Cerevisiae* cell cycle expression data [96] obtained from `http://arep.med.harvard.edu/biclustering/`, and the *Colon Cancer* data [97] obtained from `http://microarray.princeton.edu/oncology/` . All experiments were performed using the linear manifold algorithm (LMCLUS) presented in chapter 8.

10.4.1 Yeast Data

The yeast data contains 2884 genes (objects) and 17 time points/conditions (dimensions), and is a very attractive data set for evaluating clustering algorithms because many of the genes contained in it are biologically characterized and have already been assigned to different phases of the cell cycle. LMCLUS was applied on the yeast data in two different modes. One allowing it to detect all types of pattern clusters that are embedded in linear manifolds (one of the advantages of the linear manifold clustering paradigm over related pattern clustering methods), and the other forcing it to search only for shift pattern clusters so that the results can be compared with other reported results (most pattern clustering methods are restricted to shift patterns). Forcing the algorithm to search only for shift patterns was achieved by replacing LMCLUS's "goodness of separation" criteria with the *mean squared residue score* (MSRS) used by the biclustering algorithm [67] and discussed in section 6.6.1, to assess the quality of identified clusters. By forcing the algorithm to detect only

shift pattern clusters LMCLUS detected 98 clusters. We compared these clusters with the 100 biclusters identified by the biclustering algorithm, which were obtained from `http://arep.med.harvard.edu/biclustering/`. Fig. 10.5 shows boxplots highlighting the differences between the two results in terms of the size, dimension, and MSRS of the clusters detected. LMCLUS detected smaller clusters (maximum of about 150 genes per cluster as opposed to 1000), which biologically makes sense, since the whole yeast genome contains roughly only 6000 genes, and typical functional categories of the yeast genome contain dozens rather than hundreds of genes. LMCLUS also found clusters in higher dimensions (as low as 10), which is biologically more significant. Generally speaking, this is because coherent behavior patterns observed over more dimensions may provide stronger evidence that the objects under consideration are related, and from a statistical point of view is less likely to occur by chance. Finally, the median MSRS of the clusters detected by LMCLUS was slightly larger, but it was able to find many clusters with much smaller MSRS than the ones found by the biclustering algorithm. We note that the authors of the biclustering algorithm used MSRS=300 as a threshold to qualify "worthy" biclusters.

We also evaluated the biological significance of the clusters LMCLUS produced by means of *function enrichment* [96]–the degree to which the clusters grouped genes of common function. This was done by computing for each cluster P-values (using the hypergeometric distribution) of observing a certain number of genes within a cluster from a particular MIPS (Munich Information Center for Protein Sequences) functional category found at `http://mips.gsf.de/`. Some of the clusters demonstrated significant grouping (very small P-values) of genes within the same functional class. Table 10.1 shows three of the more significant clusters found by LMCLUS.

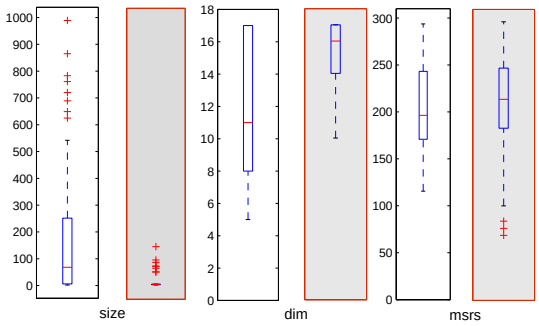


Figure 10.5: Boxplots comparing the characteristics of biclusters obtained by the biclustering algorithm and linear manifold clusters (gray boxs) of the yeast genome.

Genes in Cluster	MIPS Functional Category	Genes in Category	Clustered Genes	P-value
68	ribosome biogenesis	215	49	<1e-14
	protein synthesis	359	52	<1e-14
	cytoplasm	554	54	<1e-14
7	endoplasmic reticulum	157	4	1.251e-05
12	DNA processing	251	6	2.934e-06
	cell cycle and DNA processing	628	8	3.345e-06

Table 10.1: MIPS gene function enrichment.

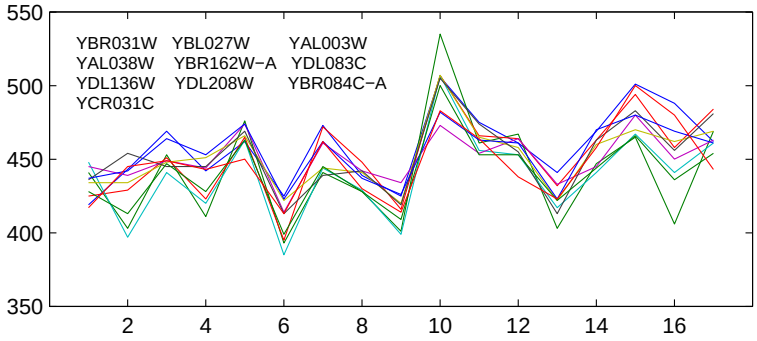


Figure 10.6: A yeast cluster detected by LMCLUS which manifests a scaling pattern and negative correlations.

Allowing LMCLUS to consider all types of patterns, or all linear manifolds, it detected 45 clusters of generally low dimensionality, and which induced different types of patterns. Fig. 10.6 shows one such cluster of ten genes embedded in a 3-dimensional linear manifold, and which exhibits a scaling pattern and negative correlations. Among these genes, six (YBL027W, YBR031W, YBR084C-A, YCR031C, YDL083C, YDL136W) enriched the ribosome biogenesis functional category with a P-value of 2.848e-07.

10.4.2 Cancer Data

The cancer data contains 2000 genes (objects) and 62 tissue samples (dimensions), of which 40 were colon tumor and 22 normal colon samples. The goal in this experiment was to identify gene clusters that can differentiate the cancerous tissues from the normal ones. These clusters may later afford researchers the ability design classifiers for diagnostic purposes. LMCLUS identified one such cluster containing 229 genes expressed in 12 cancerous tissues and no normal tissues. LMCLUS also found one

cluster containing 44 genes which were expressed in all 62 tissues, indicting that these genes cannot be used to differentiate the normal from the cancerous tissues. The rest of the clusters contained a portion of the samples but none with an overwhelmingly majority of normal or cancerous tissues.

10.4.3 Notes

Until a consensus is reached pertaining to the question of which patterns carry biologically relevant information, and appropriate cluster validation techniques are devised, the clustering of gene expression data can be considered a subjective process which depends on the cluster models assumed in advance, and which yields different views or interpretations of the data depending on the clustering technique fitted to the assumed cluster models. In this respect the linear manifold clustering paradigm makes less assumptions as it generalizes several more specific cluster models. Furthermore, the derivation and evaluation of validity measures appropriate for gene expression data is beyond the scope of this thesis. Hence, the experimental results presented in this section are by no means an attempt to report discoveries of biologically relevant and previously unknown results. The aim is to merely point out and demonstrate a method which is based on the concept of linear manifolds and which is able to identify pattern clusters that could be of biological relevance, some of which are overlooked by other methods.

We also note that one of the drawbacks of LMCLUS in its proposed form is that when applied to the application of pattern clustering it requires post-processing, designed to extract the relevant features (subspaces) of the pattern clusters it identifies. LMCLUS in its presented form does not distinguish the relevant features from the irrelevant ones during the clustering process, since pattern clusters even when present

only in subspaces still appear as linear manifolds in the full space, as illustrated in section 10.2. Putting it in other words, pattern clusters when embedded in subspaces from lines, the irrelevant dimensions transform these lines into linear manifolds of higher dimensionality in the full space, and LMCLUS does not give preference or distinguishes lines from higher dimensionality manifolds. It is designed to identify sets of points that fit linear manifolds that appear in the full space, whether they are lines or not. Thus, the post-processing is essentially designed to separate the line formed by the pattern, from the linear manifold's remaining $d-k$ dimensions formed by the irrelevant features. Ideally, this extraction should take place during the clustering process it self, since the evaluation of pattern clusters and hence the determination of clusters may be different and more accurate when done in subspaces as opposed to the full space. A possible solution, which is presented in chapter 11, is to design an algorithm which is able to identify lines or one-dimensional linear manifolds in subspaces rather than higher dimensional manifolds in the full space. Nonetheless, knowing that pattern clusters induce large correlations among their constituting features, we extracted relevant features of pattern clusters discovered by LMCLUS during the post-processing stage by selecting the k most correlated features. Because the dimensionality $l = d - k + 1$ of the each linear manifold was output along with each cluster, the number k of features that needed to be selected in this process was determined by $k = d - l + 1$.

Chapter 11

The Line Clustering Algorithm

11.1 Introduction

In chapter 10 we showed that pattern clusters and correlations can be expressed as linear manifolds, and therefore the problem of searching for pattern or correlation clusters can be cast to the problem of searching for linear manifold clusters. The advantages of this approach to the problem of pattern and correlation clustering were also discussed. Namely, as opposed to current state of the art methods, by searching for linear manifolds rather than specific patterns we can cluster concurrently all types of patterns, including those that are overlooked by existing methods, and by that allow for less biased results which are more truthful to the data.

Among the insights discussed in chapter 10, we showed that within the subspace of relevant features pattern clusters (shift and scaling) manifest themselves as one-dimensional linear manifolds or lines, and in the full space as higher-dimensional linear manifolds. Furthermore, it was shown that pattern clusters give rise only to specific instantiations of correlations and linear dependencies, and that more complex linear dependencies or correlations can be captured by one-dimensional linear manifolds. More specifically, proposition 10.3.2 shows that a set of points induce large

correlations (both positive and negative) among a set of features if and only if they fit a one-dimensional linear manifold. Hence, the problem of pattern or correlation clustering can be recast to the problem of searching in subspaces of the data for groups of points that are embedded in one-dimensional linear manifolds, rather than higher dimensional linear manifolds in the full space.

Using the linear manifold clustering algorithm - LMCLUS, we presented a series of experiments in section 10.4 demonstrating the applicability of the linear manifold clustering paradigm to the problem of pattern and correlation clustering. While effective in identifying pattern clusters and correlations, we discovered several problems associated with LMCLUS that render it from truly exploiting the full potential that linear manifolds offer to the problem of pattern and correlation clustering. One of the problems was discussed in section 10.4.3. In the following we reiterate the problem and discuss two additional ones.

- LMCLUS requires post-processing used to extract the relevant features (subspaces) of the pattern clusters it identifies. It outputs linear manifolds in the full space whose dimensions include both the relevant and irrelevant dimensions (features) of a pattern cluster, which must be separated. Ideally this process should take place during the clustering itself and not after, since the evaluation of pattern clusters and hence the determination of clusters may be different and more accurate when done in subspaces as opposed to the full space. LMCLUS in its proposed form is not designed to search for linear manifolds in subspaces of the data but only in the full space. Some solutions such as the one discussed in section 10.4.3 can be incorporated into the search, but they are computationally expensive (e.g., performing PCA) and not always successful (do not yield

the correct dimensions).

- LMCLUS tends to find arbitrarily oriented linear manifolds and not necessarily axis-parallel ones, as they typically fit the data better. As mentioned, pattern clusters are defined in terms of the original measurement features, and manifest themselves as axis-parallel linear manifolds in the full space. Hence, many of the clusters returned by LMCLUS may actually not be pattern clusters if arbitrarily oriented linear manifolds are more true to the data. It is possible to force LMCLUS to search for axis-parallel manifolds by modifying its criterion function, as discussed in section 10.4.1. However these modifications will not be natural to the algorithm, possibly requiring other adjustments which are computationally expensive, and may only be able to identify specific patterns such as MSRS used to identify shift pattern clusters.

- LMCLUS is designed to identify lower-dimensional and not higher-dimensional linear manifolds. When trying to identify higher-dimensional manifolds, LMCLUS may suffer from the lack of sufficient enough *degrees-of-freedom*. That is, when searching for higher dimensional linear manifolds more points, that are not always available, need to be sampled to instantiate the linear manifold model and even more to substantiate it. As mentioned the dimensionality of a linear manifold corresponding to a pattern cluster embedded in a k-dimensional subspace is $d - k + 1$. In many gene expression experiments the data dimensionality is large (e.g., $d = 100$ dimensions) and the patterns are embedded in relatively lower-dimensional subspaces (e.g., $k = 20$ dimensions). Thus, in order to identify these pattern clusters LMCLUS must search for higher-dimensional linear manifolds (e.g., 100-20+1=81-dimensional linear manifolds), and to do

so needs larger point samples to instantiate a manifold (e.g. 81+1=82 points) which may not always be available (e.g., only 70 points are to be clustered). Even when there are enough points to instantiate a manifold, there might not be enough points to substantiate it, i.e., validate that it is meaningful by observing that many points indeed lie on it relative to other sets of points that do not. Moreover, the computational overhead associated with higher-dimensional linear manifold clustering may make LMCLUS impractical, as discussed in section 8.7.

In this chapter we present a line clustering algorithm as an alternative to LMCLUS, which is based on a one-dimensional linear manifold cluster model and on proposition 10.3.2, and which is specifically targeted at pattern and correlation clustering while rectifying the problems associated with LMCLUS. We strongly emphasize that the line clustering algorithm by no means undermines the capabilities of LMCLUS. As a more generic method LMCLUS applies to wider range of applications, including pattern and correlation clustering, which were covered throughout this thesis, while the line clustering algorithm is a fine-tuned method which is only applicable to the problem of pattern and correlation clustering.

11.2 Overview

Supported by proposition 10.3.2 the problem of searching for groups of points which induce large correlations in subspaces (subsets of features/dimensions) of the data is cast into the problem of searching for line clusters embedded in subspaces of the data. We reiterate, that by searching for correlations, pattern clusters as a special case of correlations can also be identified. The line clustering algorithm called SLCLUS

(Subspace Line CLUStering) searches for line clusters which may be embedded in axis-parallel subspaces of the data. Similar to LMCLUS it is based on a stochastic model fitting technique, but takes on a slightly different approach. It uses a line detector procedure to search for line clusters in subspaces of the data, a *forward-feature-selection* technique to extend and refine a cluster, and a *random walk* on the feature's lattice to select an initial set of features on which the clustering process is initiated. The algorithm in its most generic form can be stated as follows: find the "best" line cluster in an initial set of dimensions using a random walk on the feature's lattice. Using forward-feature-selection add dimensions (features) to extend and refine the line cluster until no more dimensions can be added, in which case a line cluster is assumed to be found. Remove the identified line cluster from data set and reapply the first two steps on remaining set of points. The algorithm is designed so that it detects the the largest possible clusters embedded in the largest possible subspaces. The rational as mentioned before is that correlations induced by larger clusters in a larger set of features provide stronger evidence pertaining to the relationship between the objects under consideration, and from a statistical point of view is less likely to occur by chance. The effectiveness and advantages of the algorithm as a pattern and correlation clustering method is demonstrated by a set of experiments at the end of the chapter.

11.3 The Line Cluster Model

In this section we present a line cluster model that is a slight variation of the linear manifold cluster model in that it only considers one-dimensional linear manifolds, and assumes a slightly different distribution of points off the manifold, which is in this

case referred to as the distribution of the *error* associated with each point.

Suppose a line cluster X exists in a k-dimensional axis-parallel subspace. Let $\mathbf{x}$ be a $k \times 1$ vector representing some point in X, $\boldsymbol{\beta}$ be a unit norm $k \times 1$ vector that spans a $1D$ subspace, $\overline{\boldsymbol{\beta}}$ be a $k \times k-1$ matrix whose $k-1$ column vectors form an orthonormal basis that spans the space orthogonal to the space spanned by $\boldsymbol{\beta}$.

Definition 11.3.1 (The Line Cluster Model)**.** *Let $\boldsymbol{\mu}$ be some point in $\mathbb{R}^k$, ϕ be a zero mean random scalar distributed according to $U(-R/2, +R/2)$ where R is the range of the data, and $\boldsymbol{\epsilon}$ be a $k-1 \times 1$ random vector distributed according to $N(\mathbf{0}, \sigma^2 I)$, where $\sigma \ll R$. If X is a line cluster, then each $\mathbf{x} \in X$ is modeled by,*

$$\mathbf{x} = \boldsymbol{\mu} + \boldsymbol{\beta}\phi + \overline{\boldsymbol{\beta}}\boldsymbol{\epsilon}. \tag{11.3.1}$$

Explanation: similar to the linear manifold cluster model, the idea is that each point in a cluster lies close to a line (1-dimensional linear manifold) of finite extent, which is defined by $\boldsymbol{\mu}$, a translation vector, the space spanned by the vector $\boldsymbol{\beta}$, and the range parameter R. Since $\mathrm{E}[\phi] = 0$ and $\mathrm{E}[\boldsymbol{\epsilon}] = \mathbf{0}$, the mean of the cluster just as with linear manifold clusters is $\boldsymbol{\mu}$. On the line the points are assumed to be uniformly distributed in direction $\boldsymbol{\beta}$ according to $U(-R/2, +R/2)$, but again this assumption is not binding, and can be replaced by any other distribution with symmetric support. ϕ in this case can be viewed as the displacement or distance of a point from the line's center. The third component models a small random error associated with each point on the line. The idea is that each point may be perturbed in directions that are orthogonal to the subspace spanned by $\boldsymbol{\beta}$, modeled by requiring that $\boldsymbol{\epsilon}$ be a $(k-1) \times 1$ random vector, normally distributed according to $N(\mathbf{0}, \sigma^2 I)$, where σ is much smaller than R. As opposed to the linear manifold cluster model, the error has a diagonal covariance matrix with equal variances σ^2 along the diagonal. This assumption is common to other quantitative models such as the shift, scaling, and

regression models that assume homoscedasticity (constant variance). The assumption also simplifies the model making statistical inferences (done later) easier. The error ϵ in this case, can be thought of as the displacement or distance of a point to its projection onto the line. The addition of the error term essentially transforms the line into a thin elongated cylinder.

11.4 Line Detectors

Four line detectors were evaluated amongst which the most accurate and efficient was chosen to be used by SLCLUS; a RANSAC [98] based method, two versions of LMCLUS, and an adaptation of K-means [12] to line clustering, all of which are stochastic (based on sampling). RANSAC (Random Sample Consensus) is a model fitting algorithm like LMCLUS that repeatedly samples points from the data to instantiate the model parameters (e.g., two points that may a define a line), finds data points that fit the model, and returns the best model along with its fitted points. The two versions of LMCLUS we used as part of our evaluation were forced to search only for one-dimensional manifolds, one returning the first identified manifold, and the other the "best" among all the one-dimensional manifolds identified. The K-means adaptation we used proceeded as normal, but instead of recomputing cluster means it recomputes the line that best fits each cluster, and returns the best cluster. Each of these algorithms were evaluated using different criteria to qualify the "best" clusters or models.

The *Hough transform* [99] which may also be used as a line detector requires the quantization of the "parameter space" into bins, and selects the bins with the largest count corresponding to a set of parameters which define a model that passes through

Algorithm LineDetector $(X,\ s,\ d,\ t)$
  repeat s times
    sample two points $\mathbf{x}_1, \mathbf{x}_2$ from X
    determine line parameters $\boldsymbol{\mu} = \mathbf{x}_1,\ \boldsymbol{\beta} = (\mathbf{x}_2 - \mathbf{x}_1)/\|(\mathbf{x}_2 - \mathbf{x}_1)\|$
    determine the number of points in the data that are within
      distance d ("inliers") from the line
  return the line with most inliers if it includes at least t inliers

Figure 11.1: A RANSAC based line detector algorithm.

most points. In higher dimensional spaces more parameters are required to define a model (even as simple as a line model), i.e., the dimensionality of the parameter space gets larger. As a result the average number of counts per single bin is very low, making it hard to distinguish meaningful bins from others. Moreover, the quantization of the parameter space and accumulation of counts renders the method from being efficient in higher dimensional spaces. Due to these deficiencies the Hough transform was omitted from the evaluation of line detectors.

Each of the candidate line detector methods was tested on several different synthetic data sets, and its accuracy (number of points correctly classified) and efficiency (time in seconds) were measured. Because of their stochastic nature, each method was run several times on each data set, and the averages were recorded. We found LMCLUS and RANSAC to be the most accurate, and among the two RANSAC to be more efficient. A sample of these results is presented in table 11.1 for illustrative purposes. Based on this finding, in addition to RANSAC's simplicity and relatively intuitive input parameters, we chose a RANSAC based method as our line detector. The final version of the line detector algorithm is outlined in Fig. 11.1. The last step (return statement) ensures that the largest line clusters are detected by the overall line clustering algorithm.

	RANSAC		LMCLUS_b		LMCLUS_1		line K-means	
	accuracy	time	accuracy	time	accuracy	time	accuracy	time
D_1	0.95	0.05	1.00	2.68	1.00	1.79	0.56	0.05
D_2	0.95	0.16	0.69	12.2	0.41	3.26	0.58	0.40
D_3	0.84	0.16	0.92	9.46	0.90	3.44	0.57	0.53
D_4	0.93	0.35	0.91	17.2	0.93	4.13	0.62	0.78
D_5	0.92	0.48	0.83	21.9	0.87	4.41	0.72	1.06
D_6	0.91	0.89	0.61	18.1	0.64	5.03	0.73	1.62
D_7	0.78	0.10	0.95	6.19	0.95	3.08	0.67	0.22
D_8	0.91	0.16	0.86	11.9	0.84	3.77	0.72	0.32
D_9	0.81	0.16	0.81	10.1	0.83	3.37	0.70	0.33
D_{10}	0.74	0.48	0.47	29.5	0.41	4.18	0.46	0.90

Table 11.1: A summary of the line detector selection experiment. The table shows the accuracy and running time (in seconds) of each of the line detector methods applied on a sample of 10 synthetic data sets. LMCLUS_b is the version that returns the best line cluster among all identified, and LMCLUS_1 the version returning the first cluster identified.

11.5 Feature Selection

Feature selection techniques attempt to discover the most revelent attributes of the data as part of a machine learning or model building process. In our case feature selection is used to select the best and largest possible set of dimensions (features) in which a set of points fits a line. Finding the the optimal set of features by an exhaustive search trough all possible sets of features is typically infeasible. For this reason most feature selection techniques employ a greedy hill-climbing approach. The basic procedure involves identifying an initial model, and iteratively improving the model by adding or removing features in accordance with some criteria until there is no improvement or when a predetermined number of steps has been reached. *Forward selection* techniques are *bottom-up* methods which start with an empty or small set

of features and at each step add the most "relevant" feature to the model. *Backward elimination* techniques are *top-down* approaches which start with the full set of features and remove the most "irrelevant" feature at each step. *Stepwise* methods employ a combination of two, and depending on the direction of search are called *forward stepwise* and *backward stepwise* [100]. Supported by the following proposition we chose to use a bottom-up approach to extend and refine line clusters.

Proposition 11.5.1 (Downward Closure Property of Lines)**.** *If there exists a line in a set of k dimensions then there exits a line in all $k-1$ subsets of these k dimensions.*

Proof: Based on the model given in (11.3.1) and similar to the proof of proposition 10.3.2 each of the k components x_i of each point constituting a line can be modeled by $x_i = \mu_i + \beta_i \phi$ where x_i and ϕ are random variables. We can therefore select any subset of $k-1$ components where each of the components is modeled as above, and join them together into a vector producing $\mathbf{x}' = \boldsymbol{\mu}' + \boldsymbol{\beta}' \phi$, the model of a $k-1$-dimensional line. □

Proposition 11.5.1 tell us that if a set of points form a line cluster in some set of dimensions it is possible to commence the search for the cluster in a smaller set of dimensions, and iteratively extend it in a bottom-up manner using forward selection. Moreover, the search for clusters in lower dimensions is typically easier (faster) than a search for clusters in higher dimensions. Proposition 11.5.1 also provides pruning power. That is, if a line cluster is not visible in a smaller set of dimensions it is not necessary to search for it in higher dimensions. Using this property we can also devise a termination condition for the algorithm, i.e., if the line detector is not able to detect any more clusters in a small initial set of dimensions then the algorithm should terminate. The combination of the bottom-up approach and proposition 11.5.1 also

ensures that the line clusters are found in the largest possible subspaces.

Because a top-down approach will not be able to exploit the advantages of the downward closure property of lines we choose to adopt a bottom-up forward (or stepwise) selection approach. It now remains to determine how an initial set of features is selected to start the clustering process. One possibility is to start the process with line clusters of the smallest possible dimensionality, i.e., two-dimensional line clusters. This can achieved by searching through all possible 2D subspaces for a line cluster using the line detector algorithm. Needless to say that this possibility is computationally feasible as it is only quadratic in the dimensionality of the data. However, confirmed by experiments presented in section 11.12, clusters tend to overlap when projected from higher dimensional spaces into lower dimensional spaces. Hence, the projection of several line clusters embedded in higher dimensional spaces into a lower dimensional space may either mask each other or appear as a single cluster. This in turn may "confuse" feature selection used to extend the cluster, in the determination of which features are relevant to the cluster. Ideally the clustering process should start with a larger set of initial features, close in number to the dimensionality of the subspace in which some line cluster exists. This will not only improve cluster detection accuracy but also improve efficiency as the extension process will be shorter. To achieve this goal we propose a method that is based on a *random walk* on the *features lattice*.

11.6 Selecting an Initial Set of Features by Random Walk

The potential of random walk approaches in context of *level-wise* algorithms such as those frequently used to mine association rules is discussed in [101]. However, completely different from our approach, rather than walking up the itemset or features lattice the random walk used by the line clustering algorithm walks down the features lattice so that subspaces and line clusters of higher dimensionality are examined and intercepted sooner. The basic idea can be stated as follows: starting from the full set of features, randomly remove one feature at a time, after each removal invoke the line detector algorithm to detect line clusters in the subspace defined by the remaining features, and repeat the process until a line cluster is detected, or until no more features are left to be removed. As mentioned, ideally we would like the random walk to stop sooner, i.e., after the removal of fewer features, so that initial line clusters will be intercepted at higher dimensions closer to the dimensionality of the subspace in which they are embedded. This will not only improve accuracy, but also efficiency.

More formally, let F be the full set of features, and $\mathcal{L}_F$ be the lattice (poset) defined by $(\mathcal{P}(F), \subseteq)$. If F_1 and F_2 are two elements in $\mathcal{L}_F$ (subsets of F), we say that F_2 is more *general* than F_1 or F_1 more *specific* than F_2 denoted by $F_2 \prec F_1$ if $F_2 \subset F_1$. Further, let $F' \subseteq F$ be a set of features (subspace) in which some line cluster exists, and F'' be the set of features remaining after each feature removal during the walk. If at a certain point during the random walk $F'' \subseteq F'$, i.e., a subset of features that constitute a higher dimensional line cluster is detected, we can stop the random walk and use F'' as our starting point (initial set of features). Then using forward selection we can extend the line cluster currently residing in the subspace

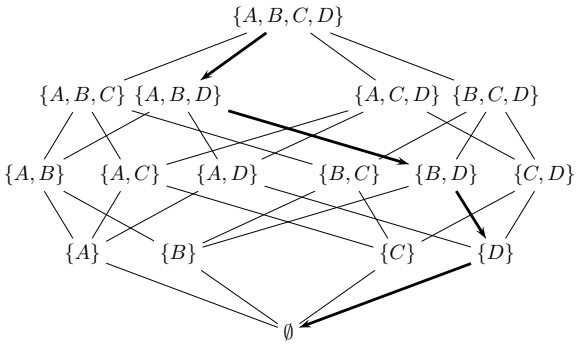


Figure 11.2: A random walk on a four-feature-lattice. The arrows show one possible walk that terminates with an empty set of features.

defined by the features of F'' to the higher dimensional subspace in which the cluster exists defined by the features of F'. Due to the downward closure property of lines, once this condition is met it is not necessary to continue the walk (remove features) as any subset of F'' will also contain a line cluster. Thus, the method can be restated as: perform a *random walk* on $\mathcal{L}_F$ starting from F and moving in the *generalization* direction until either $F'' \subseteq F'$ or $F'' = \emptyset$. This idea is illustrated by Fig. 11.2.

We now examine the statistical properties of the random walk. In order to do so we assume the line detector procedure will always be able to identify a line cluster in a subspace if such a cluster exists. Let X denote a random variable counting the number of features remaining upon termination of the random walk, i.e., the dimensionality of the subspace in which a line cluster (if one exists) was first detected. The larger X is the better. We say the random walk *fails* if it fails to detect a line cluster when one exists, i.e. when $X < 2$. Likewise we say the random walk *succeeds* when $X \geq 2$, that is, a set of at least two features that are a subset of the set of features in which a line cluster exists were encountered by the walk. Let d denote the dimensionality

of the data, k the dimensionality of the subspace in which a line cluster exists, and c the number of clusters in the data. The questions we would like to answer are the following:

Case 1: Assuming there is only one line cluster in the data.

1. What is $P(success) = P(X \geq 2|d, k)$? This will tell us how likely we are to intercept some subspace in which the cluster exits by the random walk.

2. What is $\mathrm{E}[X|d, k]$? This will tell us at which dimension or how soon we can expect to intercept the line cluster. The larger this value the better, as it will indicate that we can expect to intercept the cluster sooner.

3. Given d, what should k be so that $\mathrm{E}[X|d, k] \geq 2$? or alternatively what should the ratio between d and k be so that $\mathrm{E}[X|d, k] \geq 2$? This will tell us how large should the subspace in which the cluster exists be so that we can expect the random walk to succeed. Alternatively, what is the range of subspace dimensionalities in which the walk is effective. This will guide us to decide on which types of data sets (containing clusters in lower or higher dimensional subspaces) the random walk can be applied successfully. It is reasonable to assume that the larger the dimensionality of the subspace with respect to the dimensionality of the data the more likely it is that the walk will succeed or intercept a cluster sooner.

Case 2: We now assume that there are c and not one line cluster in the data, and that each of the clusters exists in a k-dimensional subspace, but not necessarily the same subspaces. The requirement that all the clusters reside in a k-dimensional subspace and not in subspaces of different dimensionalities is considered a limiting case. The

questions we want to answer in this case are similar to those of the first case with the exception that now there are more clusters in the data, and that $P(X = x|d, k, c)$ denotes the probability that the random walk intercepted a set F'' of x features such that F'' is a subset of at least one of the feature sets in which some of the c clusters are embedded

1. What is $P(success) = P(X \geq 2|d, k, c)$?
2. What is $\mathrm{E}[X|d, k, c]$?
3. Given d and c, what should k be so that $\mathrm{E}[X|d, k, c] \geq 2$?
4. Will the random walk perform better when more clusters exist in the data? It seems reasonable to assume that the answer is yes.

Lemma 11.6.1.

$$P(X = x|d, k) = \frac{\binom{k}{x}}{\binom{d}{x}} - \frac{\binom{k}{x+1}}{\binom{d}{x+1}}.$$

Proof: Given $|F| = d$, $|F'| = k$ and $X \leq k$. $X = x$ implies that the random walk yielded a subset of features F'' such that $F'' \subseteq F'$ and $|F''| = x$. This in turn happens when $d-x$ features have been removed (sampled) from the set F by the random walk, and of those features removed exactly $k - x$ are removed (sampled) from the set F' (to yield the set $F'' \subseteq F'$), and $d - x - (k - x) = d - k$ features removed (sampled) from the set $F - F'$ where $|F - F'| = d - k$. The probability of this event occurring, similar to other hypergeometric problems, is given by

$$\frac{\binom{k}{k-x}\binom{d-x}{d-x}}{\binom{d}{d-x}} = \frac{\binom{k}{x}}{\binom{d}{x}}.$$

However, this probability does not take into account the order in which the features are removed. It includes other successful random walks that stopped earlier, i.e.,

$X > x$ ($F'' \subseteq F'$ and $F'' > x$), but would have lead to $X = x$ if the random walks were allowed to continue removing features. After all, if $F'' \subseteq F'$ then any subset of F'' will also be a subset of F'. Therefore,

$$\frac{\binom{k}{x}}{\binom{d}{x}} = P(X \geq x|d,k).$$

Since

$$\begin{aligned} P(X = x|d,k) &= P(X \geq x|d,k) - P(X \geq (x+1)|d,k) \\ &= \frac{\binom{k}{x}}{\binom{d}{x}} - \frac{\binom{k}{x+1}}{\binom{d}{x+1}}, \end{aligned}$$

we get the final result □.

Knowing $P(X = x|d,k)$ we can now answer all the questions of case 1 (only one cluster exists in the data), by computing

$$P(success) = 1 - (P(X = 0|d,k) + P(X = 1|d,k))$$

and

$$\mathrm{E}[X|d,k] = \sum_{i=0}^{k} i \cdot P(X = i|d,k).$$

For the case of multiple clusters let $F'_1 \ldots F'_c$ denote the subsets of features in which each of the c clusters exists. Since we have no prior knowledge of the the subspaces in which the clusters exist we will assume $F'_1 \ldots F'_c$ are all drawn randomly and independently with replacement (allowing for clusters to exist in the same subspaces) from $\mathcal{L}_F$, and all of the same dimensionality (limiting case), i.e., $|F'_1| = \ldots = |F'_c| = k$. As mentioned $P(X = x|d,k,c)$ denotes the probability that the random walk intercepted a set F'' of x features, such that F'' is a subset of at least one of the F'_i's, i.e.,

$$P(X = x|d,k,c) = P\left(\bigcup_{i=1}^{c}(F'' \subseteq F'_i \cap |F''| = x)\right).$$

Lemma 11.6.2.

$$P(X = x|d, k, c) = \sum_{i=1}^{c} (-1)^{i-1} \binom{c}{i} \left[\left(\frac{\binom{k}{x}}{\binom{d}{x}} \right)^i - \left(\frac{\binom{k}{x+1}}{\binom{d}{x+1}} \right)^i \right].$$

Proof: because $F'_1, \ldots, F'_c$ are sampled independently

$$\begin{aligned} P\left(F'' \subseteq F'_1 \cap |F''| \geq x\right) = \ldots = P\left(F'' \subseteq F'_c \cap |F''| \geq x\right) \\ = P\left(F'' \subseteq F' \cap |F''| \geq x\right) = P(X \geq x|d, k) = \frac{\binom{k}{x}}{\binom{d}{x}}. \end{aligned}$$

Using the *inclusion-exclusion* principle

$$\begin{aligned} P(X \geq x|d, k, c) &= P\left(\bigcup_{i=1}^{c} (F'' \subseteq F'_i \cap |F''| \geq x) \right) \\ &= \sum_{i=1}^{c} (-1)^{i-1} \binom{c}{i} P\left(\bigcap_{j=1}^{i} \left(F'' \subseteq F'_j \cap |F''| \geq x\right) \right) \\ &= \sum_{i=1}^{c} (-1)^{i-1} \binom{c}{i} P\left(F'' \subseteq F' \cap |F''| \geq x\right)^i \\ &= \sum_{i=1}^{c} (-1)^{i-1} \binom{c}{i} P(X \geq x|d, k)^i \\ &= \sum_{i=1}^{c} (-1)^{i-1} \binom{c}{i} \left(\frac{\binom{k}{x}}{\binom{d}{x}} \right)^i, \end{aligned}$$

and once again using the identity

$$\begin{aligned} P(X = x|d, k, c) &= P(X \geq x|d, k, c) - P(X \geq (x+1)|d, k, c) \\ &= \left[\sum_{i=1}^{c} (-1)^{i-1} \binom{c}{i} \left(\frac{\binom{k}{x}}{\binom{d}{x}} \right)^i \right] - \left[\sum_{i=1}^{c} (-1)^{i-1} \binom{c}{i} \left(\frac{\binom{k}{x+1}}{\binom{d}{x+1}} \right)^i \right] \\ &= \sum_{i=1}^{c} (-1)^{i-1} \binom{c}{i} \left[\left(\frac{\binom{k}{x}}{\binom{d}{x}} \right)^i - \left(\frac{\binom{k}{x+1}}{\binom{d}{x+1}} \right)^i \right], \end{aligned} \tag{11.6.1}$$

we get the final result $\square$.

Knowing $P(X = x|d, k, c)$ by lemma 11.6.2 allows us to answer the questions of the second case (c and not one clusters exist in the data). Using Figs. 11.3-11.5 we summarize our conclusions. Figs. 11.3 and 11.4 show the change in the probability of the random walk succeeding and the expected dimensionality of the clusters intercepted by the random walk respectively, as the number of clusters in the data set and the dimensionality of the subspaces in which the clusters exist are varied. Each curve represents a different number of clusters in the data set, where the lowest curve represents a data set with one cluster and the highest with ten clusters ($c = 1, \ldots, 10$). For illustrative purposes the dimensionality of the data is fixed at 50, but similar patterns can be observed for other data dimensionalities. It is eminent from Fig. 11.3 that as the number of clusters increases and/or the dimensionality of the subspaces in which clusters are embedded is increased the probability of success increases. It is also evident that even when a small number of clusters exist in the data and the clusters are embedded in higher dimensional subspaces there is a high probability of success. Hence, one conclusion that can be drawn is that the random walk is likely to succeed when a large (not necessarily extremely large) number of clusters exist in the data and the clusters are embedded in a relatively higher dimensional subspace. Fig. 11.4 similarly shows that the random walk is more effective, that is, clusters are intercepted sooner and at higher dimensions when the clusters are embedded in higher dimensional subspaces. The more effective region of the random walk seems to be approximately the upper third range of subspace dimensionalities. The figure also shows that the expected value converges to an exponential and that the addition of more clusters beyond a certain point will not enhance the capability of the random

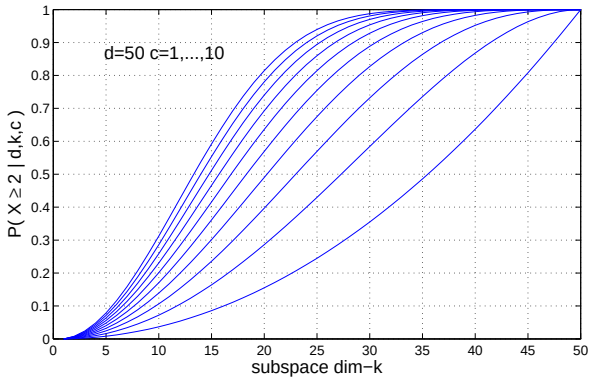


Figure 11.3: Probability of a random walk succeeding.

walk in detecting clusters sooner. Fig. 11.5 shows the dimensionality of the subspace required, given varying data dimensionalities and varying number of clusters, so that we can expect the random walk to succeed. By computing the slope of each of the curves we may conclude that beyond a certain number of clusters the required dimensionality of the subspaces in which clusters are embedded, so that on expectation the random walk will succeed, should be approximately larger than a third of the data dimensionality. One more important conclusion that can be drawn from the figures is that if the random walk fails, it is likely that the clusters are embedded in lower dimensional subspaces, in which case we can revert to our first solution, which is to initialize the clustering process by searching for two-dimensional line clusters in all possible 2D subspaces.

11.7 The Distance of a Point to a Line

The line detector algorithm requires the computation of a point's distance to a line. The squared distance (henceforth distance) of a point to a line is the norm squared of

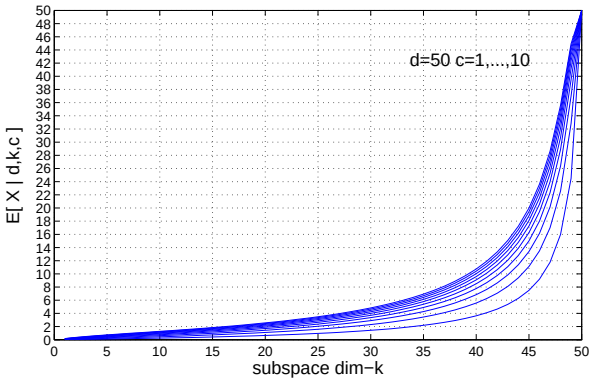


Figure 11.4: The expected dimensionality of a cluster intercepted by the random walk.

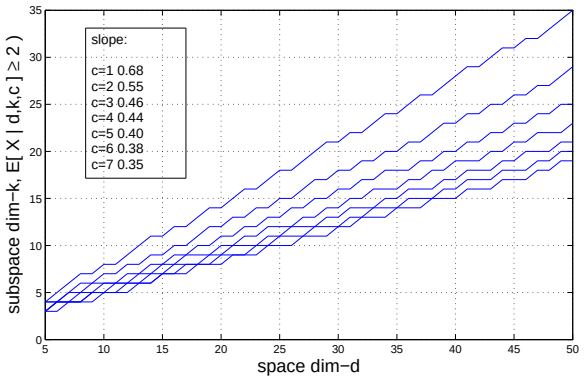


Figure 11.5: The required dimensionality of the subspace in which clusters exist so that we can expect the random walk to succeed.

its projection to the space orthogonal to the line. Formally, the distance δ of a point $\mathbf{x}$ modeled by eq. (11.3.1) to a k-dimensional line is given by:

$$\begin{aligned}
\delta &= \|(I - \boldsymbol{\beta}\boldsymbol{\beta}')(\mathbf{x} - \boldsymbol{\mu})\|^2 \\
&= \|(I - \boldsymbol{\beta}\boldsymbol{\beta}')(\boldsymbol{\beta}\phi + \bar{\boldsymbol{\beta}}\boldsymbol{\epsilon})\|^2 \\
&= \|\boldsymbol{\beta}\phi - \boldsymbol{\beta}\phi + \bar{\boldsymbol{\beta}}\boldsymbol{\epsilon} - 0\|^2 \\
&= \|\bar{\boldsymbol{\beta}}\boldsymbol{\epsilon}\|^2 \\
&= (\bar{\boldsymbol{\beta}}\boldsymbol{\epsilon})'\bar{\boldsymbol{\beta}}\boldsymbol{\epsilon} \\
&= \boldsymbol{\epsilon}'\bar{\boldsymbol{\beta}}'\bar{\boldsymbol{\beta}}\boldsymbol{\epsilon} \\
&= \boldsymbol{\epsilon}'\boldsymbol{\epsilon} \\
&= \sum_{i=1}^{k-1} \epsilon_i^2. \qquad (11.7.1)
\end{aligned}$$

According to the line cluster model $\epsilon_i \sim N(0, \sigma^2)$. Therefore the distance δ normalized by σ^2 will have

$$\frac{\delta}{\sigma^2} = \sum_{i=1}^{k-1} \frac{\epsilon_i^2}{\sigma^2} \sim \chi^2_{k-1}, \qquad (11.7.2)$$

a chi-squared distribution with $k-1$ degrees of freedom. Hence,

$$\mathrm{E}[\delta] = \mathrm{E}[\sigma^2\chi^2_{k-1}] = (k-1)\sigma^2, \qquad (11.7.3)$$

and

$$\mathrm{Var}[\delta] = \mathrm{Var}[\sigma^2\chi^2_{k-1}] = 2(k-1)\sigma^4. \qquad (11.7.4)$$

Because the distance grows with the dimensionality of the subspace in which it measured, and since the search for line clusters will be computed across different dimensionalities, we normalize the distance by its degrees of freedom $(k-1)$ or equivalently the dimensionality of the space orthogonal to the line. This creates a uniform or normalized distance measure which is independent of the dimensionality of the subspace

in which it is measured. Therefore the normalized distance $\delta/(k-1)$ has

$$\mathrm{E}\left[\frac{\delta}{k-1}\right] = \sigma^2, \tag{11.7.5}$$

and

$$\mathrm{Var}\left[\frac{\delta}{k-1}\right] = \frac{2\sigma^4}{k-1}. \tag{11.7.6}$$

The expected value and variance of the normalized distance will be used as heuristics to set the input parameters to the algorithm, discussed in section 11.10. Also note that according to lemma 8.2.1 the distance can be computed by

$$\|(I - \boldsymbol{\beta}\boldsymbol{\beta}')(\mathbf{x} - \boldsymbol{\mu})\|^2 = ||\mathbf{x} - \boldsymbol{\mu}||^2 - ||\boldsymbol{\beta}'(\mathbf{x} - \boldsymbol{\mu})||^2, \tag{11.7.7}$$

providing us with a speedup of $O(k)$, which for high dimensional spaces becomes a significant factor.

11.8 The Score (Fit) function

At each forward selection step the quality of the line cluster returned by the line detector must be assessed according to some criteria, in order to determine whether or not to proceed to the next step. The criteria we chose to use in order to assess the quality of a cluster is the "fit" of the set of points constituting the cluster to the line in which they are embedded. The fit is defined to be the average-normalized-squared-distance (average error) of the points to the line.

If k is the dimensionality of the subspace in which a cluster is detected, n the number of points constituting the cluster, X the cluster points, and $\mathbf{x}_i$ the i-th point. Then the fit or score function $J(X)$ is defined as

$$J(X) = \frac{1}{n(k-1)} \sum_{i=1}^{n} \left(||\mathbf{x}_i - \boldsymbol{\mu}||^2 - ||\boldsymbol{\beta}'(\mathbf{x}_i - \boldsymbol{\mu})||^2 \right). \tag{11.8.1}$$

Prior to the fit computation, $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$ must be estimated. $\boldsymbol{\mu}$ is estimated by computing the sample mean of the cluster. Using least-squares it can be shown that an estimate for $\boldsymbol{\beta}$ is the largest eigenvector of the cluster's covariance matrix, which can be computed using the *power method*. In section 11.11 we will discuss other methods of estimating $\boldsymbol{\beta}$.

To guide the algorithm to give certain preferences to the size and dimensionality of the cluster, $J(X)$ can be modified as follows:

$$J'(X) = J(X)n^a(k-1)^b. \tag{11.8.2}$$

For example guiding the algorithm to prefer even higher dimensional subspaces we can set b to some value less than zero. Based on eq. (11.7.5) and eq. (11.7.6)

$$\mathrm{E}[J(X)] = \sigma^2, \tag{11.8.3}$$

and

$$\mathrm{Var}[J(X)] = \frac{2\sigma^4}{n(k-1)}, \tag{11.8.4}$$

which will also be used as heuristics to set the input parameters, discussed in section 11.10.

11.9 Putting it All Together

SLCLUS (Fig. 11.6) commences by a random walk to select an initial set of features to initiate the clustering process. If the walk failed then the initial set of features is determined by searching through all possible 2D subspaces for the best 2D line cluster. If no such cluster exists then SLCLUS terminates. If either the random walk succeeds or a 2D line cluster is detected, then SLCLUS proceeds by repeatedly

calling the *forward selection* procedure (Fig. 11.7) to extend the cluster into higher dimensions and refine it, until no "improvement" can be made. A cluster is improved if its fit $J(X)$ decreased. At this point a line cluster is assumed to be found, it is outputted, removed from the data, and the algorithm is reapplied on the remaining set of points until no more points are left to be clustered. The forward selection procedure extends a cluster one dimension at a time, by choosing the dimension whose data when added to an existing cluster, attains the maximum reduction in the fit $J(X)$. We note that by calling the line detector procedure from within the forward selection procedure the algorithm ensures that line clusters are refined by also pealing off unrelated points from them. This refinement is necessary when the projection of several line clusters appear as one line cluster in lower dimensional subspaces. In addition, if the random walk fails it is likely that a cluster is embedded in a lower dimensional subspace, in which case searching through all possible 2D subspaces is likely to reveal it since it is less masked or overlapped by other cluster's projections.

11.10 Setting the Input Parameters

The algorithm requires the input of four parameters; s, a sample size parameter used to specify the maximum number of attempts that should be made by the line detector to identify a line cluster; d, the maximum distance of a point to a line allowed for it to be considered an "inlier", i.e., d can considered an error tolerance; t, a threshold used to specify that a large enough cluster has been detected; J, a "fit" threshold used to determine whether the algorithm should terminate, i.e., that no more clusters with a sufficient enough fit are left in the data.

t is relatively intuitive but may require some domain knowledge about the number

Algorithm SLCLUS (D, s, d, t, J)
  $J(X) = \infty$
  X=**perform random walk**(D)
  if $(J(X) > J)$
    X=**find the best 2D-line cluster**(D)
    if $(J(X) > J)$
      terminate
  while ($|dims(X)| < |dims(D)|$)
    X'=**ForwardSelection** (X, s, d, t)
    if $(J(X') < J(X))$
      $X = X'$
    else
      break
  output X
  $D = D - pts(X)$
  if $(D \neq \emptyset)$
    goto first step
  return

Figure 11.6: The subspace line clustering algorithm. D is the data set. X is a line cluster data structure containing the set of points in the cluster ($pts(X)$) and the set of cluster dimensions ($dims(X)$). X is updated whenever points/dimensions are added/removed from the cluster.

Algorithm ForwardSelection (X, s, d, t)
  $Y = X$
  while (unexamined dimensions of X remain)
    select an unexamined dimension of X
    add dimension data to X
    X' =**LineDetector**(X, s, d, t)
    if $(J(X') < J(Y))$
      $Y = X'$
    restore X
  return Y

Figure 11.7: The forward selection subroutine used to gradually extend the subspace in which line clusters are detected and to peel off points which do not belong to the cluster.

of points we expect to see in a cluster. For example, in gene expression clustering typical functional categories contain a small amount of genes, and therefore t should be set to a low value.

d and J are also relatively intuitive and easy to set. They pertain to the error tolerance or deviation from the line we are willing to allow, and indirectly effect the magnitude of correlations the generated clusters will induce. E.g., setting them to higher values will likely generate clusters inducing lower correlations. Assuming we are willing to accept clusters whose average deviation (distance) from a line is σ (i.e, we are assuming that $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2 I)$), then using the statistics derived in equations (11.7.5)-(11.7.6) and (11.8.3)-(11.8.4) as heuristics, we can set d and J to their expected value plus a number of their standard deviations,

$$d = \sigma^2 + c\sigma^2\sqrt{\frac{2}{(k-1)}}, \tag{11.10.1}$$

and

$$J = \sigma^2 + c\sigma^2\sqrt{\frac{2}{n(k-1)}}, \tag{11.10.2}$$

where c is the number of standard deviations. Since these values depend on k-the dimensionality of the subspace in which a cluster is either searched for or reported, the two parameters must be adjusted dynamically by the algorithm depending on k. Hence, the two parameters should simply be set to $d = J = \sigma$.

s should be selected large enough to ensure with high probability that at least one of the samples of two points are within error tolerance to a line, i.e., come from the same line cluster. This in turn will ensure that the sampled points can be used to approximate the line in which a possible cluster is embedded and to collect its remaining points. Let p be the probability that two sampled points come from the same cluster, X be a geometric random variable denoting the number of trails (samples)

K	p	s	
		$\epsilon = 0.05$	$\epsilon = 0.01$
2	0.2500	10	16
4	0.0625	46	71
6	0.0278	106	163
8	0.0156	190	292
10	0.0100	298	458
12	0.0069	430	661
14	0.0051	586	900
16	0.0039	765	1177
18	0.0031	969	1490
20	0.0025	1197	1840

Table 11.2: Values for input parameter s.

needed to get one success (two points coming from the same cluster), and $F(X)$ its cumulative distribution function. If we want to ensure with probability $1 - \epsilon$ (where $0 < \epsilon < 1$) a success then the number of trials s needed should satisfy

$$F(X) = P(X \leq s) = 1 - (1 - p)^s \geq 1 - \epsilon,$$

yielding that s should be set to

$$s \geq \frac{\log \epsilon}{\log(1 - p)}. \tag{11.10.3}$$

It now remains to determine p, the probability that two sampled points come from the same cluster. One way is to assume that there are no more than K clusters of approximately the same size in the data set, and set $p = 1/K^2$. Another way is to use t and set $p = (|D|/t)^2$, where $|D|$ is the number of points being clustered. Table 11.2 uses $p = 1/K^2$ to show some values of s, for corresponding values of K and ϵ.

11.11 Algorithmic Complexity and Optimizations

Like LMCLUS, SLCLUS is a stochastic algorithm and as such its complexity analysis is not straightforward. In the following we present asymptomatic analysis of SLCLUS's running time in terms of the number of data points, the dimensionality of the data, and the dimensionality of the subspaces in which line clusters are detected. The complexity of these three terms is not multiplicative. We also note just as with LMCLUS that the following result is only a worst case upper bound on its running time, and does not reflect its performance in practice.

Lemma 11.11.1. *The overall worst case time complexity (an upper bound) of SLCLUS in terms of the size of the data-n, the dimensionality of the data-d, the largest dimensionality of the subspaces in which line clusters are embedded-k, is* $O(n, d^3, k^3)$.

Proof: The overall complexity of SLCLUS depends the complexity of the *line detector* procedure, on the number of calls to it by the *forward selection* procedure, the complexity of the procedures used to initiate the clustering process (finding the best 2D line cluster, and the random walk), and the complexity of estimating the model parameters used to compute the score (fit).

The complexity of the line detector amounts to the complexity of computing distances of points to a line. As suggested by eq. 11.7.7 the complexity of computing a distance amounts to a one-dimensional projection whose complexity is $O(k)$, where k is the dimensionality of the subspace in which the distance is computed. Since the sample size s required by the procedure is constant, the overall complexity of the line detector procedure is $O(nk)$, where n is the number of points being examined by the procedure.

To compute the score we first need to estimate the model parameters of the line fitted to a set of points. The complexity of estimating $\boldsymbol{\mu}$ is $O(n)$, and the complexity of

estimating $\boldsymbol{\beta}$ is $O(nk^2+k^2)$, where nk^2 operations are used to compute the covariance matrix of the points, and k^2 operation are used to compute the leading eigenvector of the covariance matrix using the power method. Once these parameters are estimated, computing the score requires computing the distances of the points to the line whose complexity as before is $O(nk)$. Therefore the total complexity of computing the score is $O(n + nk^2 + k^2 + nk) = O(nk^2)$.

Searching for the best 2D line cluster requires going through all 2D subspaces $O(d^2)$, executing the line detector $O(2n)$ $(k = 2)$ and computing the score for each line detected $O(4n)$. Therefore the overall complexity of this step is $O(d^2(2n+4n)) = O(d^2n)$.

In the worst case the random walk will need to execute the line detector and compute a score for each possible subspace dimensionality, i.e., $k = 2, \ldots, d$. Hence the complexity of the random walk is $\sum_{k=2}^{d} O(nk) + O(nk^2) = O(nd^3)$.

Each dimension is examined by the forward selection procedure by calling the line detector and computing a score, with the corresponding dimension data added to the cluster. Assuming the dimensionality of the cluster before the extension is $i - 1$, then there are $d - i + 1$ dimensions to examine. Thus, the number of operations executed by this procedure is $(d - i + 1)(ni + ni^2) = O(dni^2)$. Assuming a line cluster is embedded in a k-dimensional subspace, then the number of calls to the forward selection procedure in the worst case is $k-2$, and therefore the complexity of extending a cluster from a 2D subspace to a kD subspace is $\sum_{i}^{k} O(dni^2) = O(dnk^3)$.

The total complexity of the algorithm is therefore given by

$$O(d^2n) + O(nd^3) + O(dnk^3) = O(n, d^3, k^3).$$

The flexibility of SLCLUS comes at a price. Its main bottleneck is associated with

the estimation of the line parameter $\boldsymbol{\beta}$ which is used to compute the the fit $J(X)$. Rather than estimating it by computing a covariance matrix and its associated largest eigenvector using the power method, it is possible to obtain slightly lesser accurate estimates of it by other methods. One possible method is to use the two points sampled by the line detector as a less accurate estimate. That is, along with the line cluster points, the line detector procedure also returns the corresponding parameters $\boldsymbol{\mu} = \mathbf{x}_1$ and $\boldsymbol{\beta} = (\mathbf{x}_1 - \mathbf{x}_2)/\|\mathbf{x}_1 - \mathbf{x}_2\|$ that were used to identify the line. This will result in an algorithm whose complexity is $O(n, d^2, k^2)$. Another more accurate estimate of $\boldsymbol{\beta}$ can be obtained using a monte carlo approach similar to the one used by the line detector procedure. After a line is detected, using only the line points, we repeat s times: sample two points $\mathbf{x}_1, \mathbf{x}_2$, let $\boldsymbol{\mu} = x_1$, and let $\boldsymbol{\beta} = (\mathbf{x}_1 - \mathbf{x}_2)/\|\mathbf{x}_1 - \mathbf{x}_2\|$, compute $J(X)$ the fit, and return the $\boldsymbol{\beta}$ that yielded the best fit, i.e., smallest value of $J(X)$. The rational is that because we are using only the line points it is likely (can be shown mathematically) that with a large enough sample size we will be able to sample two points that can a define a line which is very close to the true line that passes through the data. Also because the true $\boldsymbol{\beta}$ is obtained through a least squares method that essentially seeks to minimize $J(X)$, the smaller it is, the closer the corresponding sampled $\boldsymbol{\beta}$ is to its true value. The resulting complexity of the algorithm using this method is the same as the previous $O(n, d^2, k^2)$. Another possible optimization is to modify the algorithm so that clusters are extended in a bottom-up way without giving any preference to specific features. That is, features are added in any order, not necessarily the best ones first, as long as they are consistent with the model, i.e., induce a score $J(X)$ that is less than the threshold J. This will result in an algorithm (using the original method to estimate $\boldsymbol{\beta}$) whose complexity is $O(nd^3)$

which is independent of the dimensionality in which the clusters exist. Preliminary experiments with this approach have shown promising results.

We note that unlike related bottom-up subspace clustering algorithms such as CLIQUE [28] and MAFIA [50], which are exponential in the dimensionality of the subspaces in which clusters are embedded, SLCLUS without any optimizations is still only cubic. However, SLCLUS unlike these methods does not scale well with the dimensionality of the data. Nonetheless, many subspace clustering methods, particularly the top-down ones such as ORCLUS [54] which rely on eigen-decomposition are also cubic in the dimensionality of the data.

11.12 Empirical validation

We experimented with SLCLUS by applying it on both real and synthetic data sets. The algorithm was implemented in a Linux based Matlab environment.

11.12.1 Experiments with Synthetic Data

To better understand the algorithm, several dozen data sets were generated according to the line cluster model specified in eq. (11.3.1) using a similar method to the one used to generate general linear manifold clusters (section 8.8.1). The aim of the experiment with the synthetic data was to evaluate the algorithm's accuracy in the detection of both the clusters that were generated and the subspaces in which they were embedded, along with an empirical evaluation of the random walk's effectiveness and its contribution to the algorithm's performance.

To determine the algorithm's accuracy in cluster detection we used *Cluster Purity* (section 8.8.3). The determination of the algorithm's accuracy in subspace detection

was measured by the degree of correspondence between the subspace feature labels of the output and input clusters. Both accuracy measures range in $[0, 1]$, where larger values indicate better accuracy. Due to the stochastic nature of the algorithm each test was repeated between 10-100 times depending on the size and dimensionality of the data, and average accuracies were recorded.

For our first set of experiments with synthetic data we generated low-dimensional data sets in order to avoid as much as possible the overlap of cluster projections in lower dimensional subspaces. This in turn enabled us to examine the general performance of the algorithm without the random walk (the clustering process starts with clusters in 2D subspaces). These data sets consisted of 5-50 dimensions, where for each selected dimensionality three types of data sets were created. One with a small number of clusters (4-6 clusters), one with a small number of clusters to which noise points were added representing approximately 30% of the data. And a third type of data set containing a larger number of clusters (8-11). We also ensured that the dimensionality of the subspaces in which clusters were embedded ranged across the dimensionality of the data. From a pool of dozens of synthetic data sets on which the algorithm was applied, a representative sample of seven of each of the three different types (21 in total) of data sets was selected for illustrative purposes. The performance (accuracy) of the algorithm applied on these data sets is summarized in table 11.3. The table shows that the algorithm is able to maintain high levels of accuracy in cluster point detection (denoted by 'pts' in the table) and good overall performance in the detection of the subspaces which the clusters were embedded (denoted by 'dim' in the table). However, the table shows that as the dimension of the data sets increases, the accuracy in subspace detection deteriorates. We attribute this behavior to the

data dim	small		small+noise		large	
	pts	dim	pts	dim	pts	dim
5	0.96	0.99	0.89	0.97	0.92	0.94
8	0.99	1.0	0.94	0.99	0.98	0.99
10	0.99	0.98	0.97	0.98	0.93	0.94
15	0.97	0.93	0.96	0.93	0.97	0.91
20	0.99	0.79	0.90	0.90	0.96	0.95
30	0.94	0.96	0.82	0.95	0.96	0.91
50	0.97	0.65	0.93	0.92	0.90	0.67

Table 11.3: SLCLUS's performance, without random walk when applied on low-dimensional data.

overlap of cluster projections phenomena, i.e., the possibility that the projection of several clusters embedded in high dimensional spaces into lower dimensional spaces appear as single clusters when the algorithm commences the search, which "confuses" it in the determination of which features are relevant to each cluster. The next two sets of experiments demonstrate that the addition of the random walk as a mechanism of selecting a "better" initial set of dimensions which are used to start the clustering process, is able to rectify this problem.

For the second set of experiments with synthetic data we generated higher dimensional data sets, and conforming with the conclusions made in section 11.6, embedded the line clusters in subspaces whose dimensionality ranged in the upper third range of subspace dimensionalities. For example, for a data set whose dimensionality is 60 we embedded line clusters in subspaces whose range of dimensionalities was $[40, 60]$. On each of the generated data sets we applied SLCLUS with and without the random walk, in order to evaluate the random walk's utility. A sample of five data sets containing between 3-6 clusters each and whose data dimensionality ranged in 60-100 were selected for illustrative purposes. The performance of SLCLUS applied on these

data dim	-RW		+RW	
	pts	dim	pts	dim
60	0.66	0.50	1.00	0.94
75	0.66	0.56	1.00	0.86
80	1.00	0.66	1.00	0.99
90	0.79	0.41	1.00	0.93
100	0.85	0.59	1.00	0.85

Table 11.4: SLCLUS's performance, with and without the random walk, applied on high-dimensional data.

data sets is summarized in table 11.4. The table shows that when the clusters are embedded in higher dimensional subspaces employing the random walk (+RW in table) provides a significant gain in performance (specially in subspace detection) over initializing the clustering process with 2D line clusters (-RW in table). The addition of the random walk in a sense provids a mechanism to minimize and in some cases completely avoid the effects of the overlap of cluster projections phenomena.

In our third and last set of experiments with synthetic data we generated lower and higher dimensional data sets, but this time embedded the line clusters in subspaces whose dimensionality ranged uniformly across the data dimensionality, resulting in data sets whose clusters were embedded in lower, moderately higher, and high dimensional subspaces of the data. The algorithm was applied on these data sets in its complete form, i.e., with the random walk. The aim was to evaluate overall performance when no prior knowledge about subspace dimensionalities is available, which is typically the case in real experiments. In these cases it is reasonable to assume that the dimensionalities are uniformly distributed across the data dimensionality. The performance of the algorithm applied on a representative sample of these type of data sets is summarized in table 11.5. The table shows that the algorithm is able to

data dim	+RW	
	pts	dim
40	1.00	0.85
55	0.99	0.80
70	0.95	0.72
85	0.99	0.75
100	0.99	0.79
120	1.00	0.85

Table 11.5: SLCLUS's performance applied on data sets with uniformly distributed subspace dimensionalities.

achieve high accuracy in cluster point detection, and moderately high accuracy (at an acceptable level) in subspace detection. Combined with the results of the second set of experiments we validate our conclusion from section 11.6 that the random walk is more effective when the clusters are embedded in higher-dimensional subspaces.

In summary, we have demonstrated that in a controlled environment the algorithm is able to perform very well in many cases. It was able to achieve very high accuracy in cluster point detection for all cases, and relatively high accuracy in subspace detection for most cases. In the worst case the algorithm miss-detected on average only 20% of subspace dimensions. We also demonstrated that the addition of the random walk as a method of selecting an initial set of features to start the clustering process significantly enhances algorithm performance for high dimensional data sets. It also evident that in this setting it is not possible to completely avoid the effects of the overlap of cluster projections phenomena, which seem to effect only the detection of cluster subspace dimensions.

Although not designed to target the same clusters as our line clustering algorithm,

we have also implemented the FLOC algorithm [68] (a more efficient implementation of the biclustering algorithm [67]) as a representative of the pattern clustering/biclustering family of algorithms. However rather than applying it on data that follows the line cluster model, we generated synthetic data that follows the bicluster or shift pattern cluster model, which as stated earlier is a special case of the line cluster model (lines with slope one). These data sets were of varying dimensionality (low and high dimensional data) and included clusters that were embedded in subspaces whose dimensionality ranged across the dimensionality of the data. We then applied both algorithms on these data sets and compared their performance. Even though the FLOC algorithm takes as input the number of clusters assumed to exist in the data, an advantage when comparing algorithms in a supervised or controlled environment, the line clustering algorithm was able to outperform the FLOC algorithm by an average margin of 0.2 in the accuracy of cluster point detection and an average margin of 0.5 in accuracy of subspace detection. We note however that the FLOC algorithm was able to complete the experiments faster than the line clustering algorithm, a result consistent with the computational complexity of the algorithms.

We also note that a version of SLCLUS that employed a forward-stepwise feature selection approach was implemented and evaluated. We believed that the forward-stepwise approach would result in better accuracy, but discovered that on average not many backward elimination (feature removal) steps were executed, and better accuracy was not achieved. Due to the overhead associated with the backward steps we preferred use the simpler forward selection approach.

11.12.2 Experiments with Real Data

We conducted extensive tests on three real data sets. The two gene expression data sets that were used in section 10.4 – the *yeast* and *Colon Cancer* data sets, and a third one – *Jester*, which is a data set used in an online joke collaborative filtering/recommender system [102]. This data set was obtained from `http://ieor.berkeley.edu/~goldberg/jester-data/`

Yeast Data

Applied on this data set SLCLUS discovered 62 line clusters. Similar to the experiment described in section 10.4.1, we compared these clusters with the 100 biclusters reported by the biclustering algorithm. The clusters' size detected by SLCLUS ranged in $[15, 255]$ and on average much smaller than the clusters reported by the biclustering algorithm. As mentioned already, from a biological standpoint this makes sense, as the whole yeast genome contains roughly only 6000 genes, and typical functional categories of the yeast genome contain dozens rather than hundreds of genes that were included in some of the biclusters. The dimensionality of the clusters detected by our algorithm ranged in $[3, 16]$ and was on average smaller than the dimensionality of the biclusters. We also compared the *mean squared residue score* (MSRS). SLCLUS detected on average clusters with a slightly larger MSRS. However, this is reasonable since our algorithm was not restricted to searching only for shift pattern clusters for which this score was designed. Nonetheless, our algorithm was successful in finding clusters that induce large correlations. We used *average correlation*, defined to be the average of the absolute value of the correlation coefficient between each pair of features belonging to a cluster, to quantify the degree of correlation induced by a

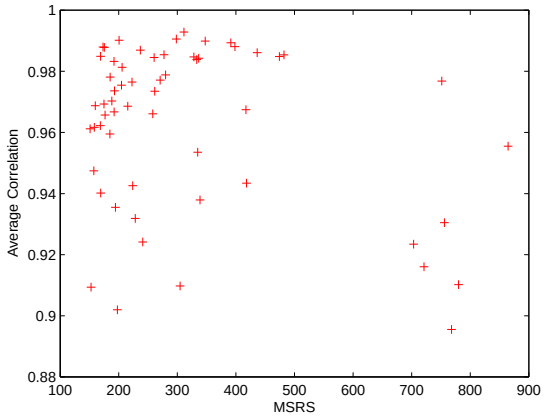


Figure 11.8: mean squared residue score (MSRS) versus average correlation of yeast clusters detected by SLCLUS.

cluster. After the removal of 3 outlier clusters (clusters with average correlation less than 0.8) the mean average correlation was found to be 0.96. Fig. 11.8 is a plot of MSRS versus average correlation of the clusters detected by our algorithm. The figure shows that there are gene clusters in the data that induce large correlations and may be functionally related, yet may not follow the shift pattern cluster model, supporting one of the main motivations for this work. We note that some of the clusters found by our algorithm did follow the shift pattern cluster model.

As in the experiment described in section 10.4.1, we also evaluated the biological significance of the clusters our algorithm produced by means of *function enrichment* – the degree to which the clusters grouped genes of common function. This was done by computing for each cluster P-values (using the hypergeometric distribution) of observing a certain number of genes within a cluster from a particular MIPS functional category. Some of the clusters demonstrated significant grouping (very small P-values) of genes within the same functional class. Table 11.6 shows five of them that had

Genes in Cluster	MIPS Functional Category	Genes in Category	Clustered Genes	P-value
211	ribosome biogenesis	215	27	1.698e-09
	protein synthesis	359	35	5.649e-09
	cytoplasm	554	37	2.696e-05
17	cytoplasm	554	15	1.565e-14
	protein synthesis	359	13	1.178e-13
	ribosome biogenesis	215	11	6.229e-13
	subcellular localisation	2256	16	8.658e-07
193	amino acid biosynthesis	118	13	5.462e-05
49	amino acid metabolism	204	6	6.740e-06
16	cell cycle and DNA processing	628	9	5.772e-06

Table 11.6: MIPS gene function enrichment.

smaller P-values. We also note that SLCLUS found more and larger clusters than LMCLUS with significant grouping.

Cancer Data

Applied on this data set SLCLUS detected 81 line clusters. The size of the clusters was generally small, the largest cluster contained 21 genes. The dimensionality of the subspaces in which the clusters were embedded ranged in $[4, 16]$. The average MSRS of the clusters was 15243, indicating that most of the clusters did not follow the shift pattern cluster model. However, their mean average correlation (after removal of two outlier clusters) on the other hand was around 0.83. Again indicating that related groups of genes may exist in the data, yet are overlooked by most clustering methods.

We also found seven gene clusters that were present in either only the normal tissues or only the cancerous tissues. One cluster contained only normal tissues and the remaining six only cancerous tissues. They contained a small number of

genes 16-18, and were embedded in subspaces of dimensionality ranging in $[4, 9]$, i.e., contained between 4-9 tissues. The average correlation of these clusters was around 0.88, higher than the rest of the clusters, providing evidence that the genes within these clusters may be functionally related and used for discriminatory purposes. Most of the remaining clusters contained a mixture of tissues none with an overwhelmingly majority of normal or cancerous tissues. We note that LMCLUS was able to find only one differentiating cluster.

Jester Data

The Jester data set contains a million continuous ratings in the range $[-10.00, 10.00]$ of 100 jokes (dimensions) from 73421 users (objects). Not all users rated all jokes, thus the data contains missing values. For our experiments we extracted a subset of 6916 users that rated all jokes. Applied on the Jester data SLCLUS identified 252 line clusters with the following statistics. The average size (number of users) of the clusters was 28 and the average dimensionality (number of jokes) of the clusters was 10. Again we examined the MSRS of the clusters to get an idea of what types of patterns were identified. Surprisingly we discovered that most of the clusters followed the shift pattern (slope one clusters in the collaborative filtering literature) with a very low average MSRS of approximately 5. This finding demonstrates that while being able to identify a wider and more general spectrum of patterns SLCLUS does not over look the more common patterns, which as stated are a special case of the line cluster model, when those are present or more evident in the data. After the removal of 29 outlier clusters the average correlation was found to be approximately 0.7, and not as high as in the preceding two experiments.

We complete the discussion of this set of experiments by reiterating the comment

about cluster validation that was made in section 10.4.3.

Conclusions

A new paradigm of clustering called "Linear Manifold Clustering" which is based on linear manifolds was designed, analyzed, and evaluated throughout this thesis. The "birth" of this paradigm of clustering is a consequence of what we believe is a need for an important yet overlooked cluster model, and as a result of what we identified as an acute need to sufficiently address certain clustering requirements that current state of art methods are unable to address. In many problem domains it assumed successfully that linear models are sufficient enough to describe and capture the data's inherent structure. Yet very few remote attempts which were typically associated with the paradigm of subspace clustering have been made to devise clustering methods able to identify or learn mixtures of linear manifolds. None of these attempts posed the problem in a model-based statistical setting intended to model and understand the underlying "process" responsible for generating sets of points that lie on lower dimensional linear manifolds.

In this thesis we introduced a formal stochastic linear manifold cluster model. The model consists of two parts: a *deterministic* part describing the linear manifold which the points fit, and a *stochastic* part describing the distribution of points on the manifold and their deviation from the manifold. Based on this model we presented a series of results and techniques demonstrating the applicability of the linear manifold

clustering paradigm to a wide range of applications. We showed that the linear manifold cluster model is a generalization of other more common and somewhat limited cluster models. This generalization allows for less assumptions to be forced on the data, which typically yield biased results, and more freedom for the data to "speak for itself". An emphasis was put on the paradigms of pattern and correlation clustering and on the application of DNA microarray analysis, where we showed that pattern clusters or correlations manifest themselves as linear manifolds in the data space. Based on these results and the linear manifold cluster model we presented two clustering algorithms: one for clustering or learning mixtures of lower dimensional linear manifolds, and the other tailored to the application of pattern and correlation clustering and to DNA micro array analysis. The efficacy of these techniques was demonstrated by a series of experiments on synthetic and real data sets, where most of the clustering validation was done using external validation techniques. Most clustering methods focus only on the grouping aspects of clustering and lack the ability to provide any scientific content. Since lacking scientific content, clustering is only a visualization tool, and because one of the ultimate goals of clustering is not only to reveal structure but also to understand the underlying mechanism or process responsible for it, we also presented two linear manifold based modeling techniques that deliver scientific content and with which data can be described. One is based on a probabilistic density estimation model with which statistical inference such as predictions can be based upon. The other is a model which describes the linear dependencies in the data in the form of a set of linear equations.

Many validity techniques have been proposed in order to evaluate clustering quality when an external assessment is not possible. However, just like the clustering

algorithms themselves they rely on certain assumptions or heuristic criteria which may not always be appropriate. Until truly unbiased and adequate cluster validity techniques are devised, it will be hard to assess or justify the output of a clustering algorithm and determine whether or not it is just an artifact of the process, in the absence of ground truth. It has been suggested that cluster validity should be posed in a statistical setting. In future work we plan to investigate such an approach, where initial ideas have already been laid out. Unlike existing methods composed primarily of one step, we believe that the validation process should be a three-step iterative process that is reapplied until a definite conclusion can be made. The first step is a statistical permutation based technique used to determine whether the cluster structure arose by chance. In the second step, also a statistical permutation based technique, we determine among a set of competing clusterings the one that best describes the underlying population and not the sample which yielded the clustering. This test is also used to determine the most appropriate cluster model or shape. Having determined that the clustering has not arised by chance, and that a certain clustering structure such as linear manifolds best describes the population, a third step that uses statistical model selection techniques such as the *Bayesian Information Criterion* may be used to determine the best clustering given a fixed clustering structure (model) and a varying set of clustering parameters.

Some optimizations to the clustering algorithms presented in this thesis have already been discussed. In future work we plan to investigate further optimizations. One particular optimization that we look forward to devise, is the approximation of the leading eigenvector of a cluster's covariance matrix which is used to estimate the model parameters of a line cluster. We conjecture that such an optimization

is possible by exploiting the unique shape of the cluster, and believe that such an optimization will enhance the algorithm's performance by a significant factor.

The linear manifold modeling technique presented in section 9.2 relied on an independence assumption which was used to simplify the modeling process by allowing the modeling of the joint distribution of points on the manifold as a product of the marginal distributions of points on each of the vectors spanning the manifold. In future work we plan to experiment with more complex modeling techniques such as graphical models that do not require an independence assumption, and that attempt to actually reveal the dependence structure on the manifold. We conjecture that such an approach will yield more accurate density estimates.

Lastly, we also plan to investigate possible extensions of the linear manifold clustering paradigm, to non-linear manifolds. One possible path we plan to explore is the use of *kernels* that will allow the detection of non-linear manifold clusters by using linear manifold clustering, but in a new coordinate system nonlinearly related to the original input space.

Bibliography

[1] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT press, 1996.

[2] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[3] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, 2nd edition*. Academic Press, 2003.

[4] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[5] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification, Second Edition*. Wiley, 2000.

[6] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.

[7] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[8] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: an efficient data clustering method for very large databases. In *ACM SIGMOD International Conference on Management of Data*, pages 103–114, 1996.

[9] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: An efficient clustering algorithm for large databases. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 73–84, 1998.

[10] George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar. CHAMELEON: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[11] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engineering*, pages 512–521, 1999.

[12] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. Berkeley, University of California Press, 1967.

[13] J. Mao and A.K. Jain. A self-organizing network for hyperellipsoidal clustering (HEC). *IEEE Transactions on Neural Networks*, (7):16–29, 1996.

[14] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, (38):293–306, 1985.

[15] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.

[16] Dan Pelleg and Andrew Moore. X-means: Extending K-means with efficient estimation of the number of clusters. In *Proceedings 17th International Conference on Machine Learning*, pages 727–734, 2000.

[17] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 144–155, 1994.

[18] Martin Ester, Hans-Peter Kriegel, Jrg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, OR, 1996.

[19] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 49–60, 1999.

[20] Alexander Hinneburg and Daniel A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pages 58–65, 1998.

[21] Jorg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.

[22] Xiaowei Xu, Martin Ester, Hans-Peter Kriegel, and Jorg Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings of the 14th International Conference on Data Engineering*, pages 324–331, 1998.

[23] Wei Wang, Jiong Yang, and Richard R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 186–195, 1997.

[24] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of the 24th International Conference on Very Large Data Bases*, pages 428–439, 1998.

[25] Daniel Barbara and Ping Chen. Using the fractal dimension to cluster datasets. In *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining*, pages 260–264, 2000.

[26] Schikuta Erich. Grid-clustering: An efficient hierarchical clustering method for very large data sets. In *Proceedings of the 13th International Conference on Pattern Recognition*, volume 2, pages 101–105, 1996.

[27] Wei Wang, Jiong Yang, and Richard Muntz. STING+: An approach to active spatial data mining. In *Fifteenth International Conference on Data Engineering*, pages 116–125, 1999.

[28] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 94–105, 1998.

[29] Bezdek J.C., Ehrlich R., and Full W. FCM: The fuzzy c-means clustering algorithm. *Computers and Geoscience*, 10(2-3):191–203, 1984.

[30] Rajesh N. Dav. Generalized fuzzy c-shells clustering and detection of circular and elliptical boundaries. *Pattern Recognition*, 25(7):713–721, 1992.

[31] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[32] Rave Harpaz and Robert Haralick. The EM algorithm as a lower bound optimization technique. Technical Report TR-2006001, Graduate Center, City University of New York, 2006.

[33] Peter Deuflhard. *Newton Methods for Nonlinear Problems Affine Invariance and Adaptive Algorithms*. Springer Series in Computational Mathematics, 2004.

[34] Peter Cheeseman and John Stutz. Bayesian classification (autoclass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*, pages 153–180. 1996.

[35] S. Wallace and D.L. Dowe. Intrinsic classification by MML – the SNOB program. In *Proceedings of the 7 Australian Joint Conference on Artificial Intelligence*, pages 37–44, 1994.

[36] P. R. Freeman C. S. Wallace. Estimation and inference by compact coding. *Journal of the Royal Statistical Society. Series B*, 49(3):240–265, 1987.

[37] C. Fraley and A. Raftery. MCLUST: Software for model-based cluster analysis. Technical Report 342, Department of Statistics, University of Washington, 1999.

[38] Anil K. Jain, Jianchang Mao, and K. M. Mohiuddin. Artificial neural networks: A tutorial. *IEEE Computer*, 29(3):31–44, 1996.

[39] Kohonen T. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[40] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, 49:291–307, 1970.

[41] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[42] R. Shamir and R. Sharan. Click: A clustering algorithm for gene expression analysis. In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology*, 2000.

[43] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[44] F. Chung. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997.

[45] Yair Weiss. Segmentation using eigenvectors: A unifying view. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 975–982, 1999.

[46] Pietro Perona and William T. Freeman. A factorization approach to grouping. In *Proceedings of the 5th European Conference on Computer Vision-Volume I*, pages 655–670, 1998.

[47] J. H. Friedman. An overview of predictive learning and function approximation. In J.H. Friedman V. Cherkassky and H. Wechsler, editors, *From Statistics to Neural Networks*, pages 1–61. Springer Verlag NATO/ASI, 1994.

[48] Lance Parsons, Ehtesham Haque, and Huan Liu. Subspace clustering for high dimensional data: a review. *SIGKDD Explor. Newsl.*, 6(1):90–105, 2004.

[49] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, 1994.

[50] Harsha S. Nagesh and Alok Choudhary Sanjay Goil. Mafia: Efficient and scalable subspace clustering for very large data sets. Technical Report 9906-010, Northwestern University, 1999.

[51] Chun-Hung Cheng, Ada Waichee Fu, and Yi Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 84–93, 1999.

[52] Bing Liu, Yiyuan Xia, and Philip S. Yu. Clustering through decision tree construction. In *Proceedings of the 9th international conference on Information Knowledge management*, pages 20–29, 2000.

[53] Charu C. Aggarwal, Joel L. Wolf, Philip S. Yu, Cecilia Procopiuc, and Jong Soo Park. Fast algorithms for projected clustering. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 61–72, 1999.

[54] Charu C. Aggarwal and Philip S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 70–81, 2000.

[55] Kyoung-Gu Woo, Jeong-Hoon Lee, Myoung-Ho Kim, and Yoon-Joon Lee. Findit: a fast and intelligent subspace clustering algorithm using dimension voting. *Information & Software Technology*, 46(4):255–271, 2004.

[56] Jerome H. FRIEDMAN and Jacqueline J. MEULMAN. Clustering objects on subsets of attributes. *Journal of the Royal Statistical Society. Series B, statistical methodology*, 66(4):815–849, 2004.

[57] Cecilia M. Procopiuc, Michael Jones, Pankaj K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 418–427, 2002.

[58] Sara C. Madeira and Arlindo L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE/ACM on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[59] J. A. Hartigan. Bdirect clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129, 1972.

[60] R. Tibshirani, T. Hastie, M. Eisen, D. Ross, D. Botstein, and P. Brown. Clustering methods for the analysis of dna microarray data. Technical report, Department of Health Research and Policy, Stanford University, 1999.

[61] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proceedings of the National Academy of Sciences*, 97(22):12079–12084, 2000.

[62] Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining (SDM'05)*, 2005.

[63] Morris H. DeGroot and Mark J. Schervish. *Probability and Statistics (3rd Edition)*. Addison Wesley, 2001.

[64] Peter Y. Chen and Paula M. Popovich. *Correlation: Parametric and Nonparametric Measures*. Sage Publications, 2002.

[65] S. Erdal, O. Ozturk, D. Armbruster, H. Ferhatosmanoglu, and W.C. Ray. A time series analysis of microarray data. In *Proceedings of the 4th IEEE Symposium on Bioinformatics and Bioengineering*, pages 366–375, 2004.

[66] Christian Böhm, Karin Kailing, Peer Kröger, and Arthur Zimek. Computing clusters of correlation connected objects. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 455–466, 2004.

[67] Y. Cheng and G. Church. Biclustering of expression data. In *International Conference on Intelligent Systems for Molecular Biology*, pages 93–103, 2000.

[68] Jiong Yang, Wei Wang, and Haixun Wang Philip Yu. δ-clusters: Capturing subspace correlation in a large data set. In *Proceedings of the 18th International Conference on Data Engineering*, pages 517–528, 2002.

[69] Haixun Wang, Wei Wang, Jiong Yang, and Philip S. Yu. Clustering by pattern similarity in large data sets. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 394–405, 2002.

[70] Jian Pei, Xiaoling Zhang, Moonjung Cho, Haixun Wang, and Philip S. Yu. Maple: A fast algorithm for maximal pattern-based clustering. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 259– 266, 2003.

[71] L. Lazzeroni L. and A. Owen. Plaid models for gene expression data. *Statistica Sinica*, 12:61–86, 2002.

[72] Kevin Y. Yip, David W. Cheung, Michael K. Ng, and Kei-Hoi Cheung. Identifying projected clusters from gene expression profiles. *Biomedical Informatics*, 37(5):345–357, 2004.

[73] Lizhuang Zhao and Mohammed J. Zaki. Microcluster: Efficient deterministic biclustering of microarray data. *IEEE Intelligent Systems*, 20(6):40–49, 2005.

[74] Jesus S. Aguilar-Ruiz. Shifting and scaling patterns from gene expression data. *Bioinformatics*, 21(20):3840–3845, 2005.

[75] Richard P. Feynman. *QED: The Strange Theory of Light and Matter*. Princeton University Press, 1985.

[76] J. Kittler and J. Illingworth. Minimum error thresholding. *Pattern Recognition*, 19(1):41–47, 1986.

[77] P. Diaconis and D. Freedman. Asymptotics of graphical projections. *The Annals of Statistics*, 12(3):793–815, 1984.

[78] Sungzoon Cho, Robert M. Haralick, and Seungku Yi. Improvement of kittler and illingworth's minimum error thresholding. *Pattern Recognition*, 22(5):609–617, 1989.

[79] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of the johnson-lindenstrauss lemma. Technical Report TR-99-006, Berkeley, CA, 1999.

[80] Alexei D. Miasnikov, Jayson E. Rome, and Robert M. Haralick. A hierarchical projection pursuit clustering algorithm. In *17th International Conference on Pattern Recognition (ICPR 2004)*, pages 268–271, 2004.

[81] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[82] S. Hettich and S. D. Bay. The UCI KDD archive. http://kdd.ics.uci.edu (1999).

[83] C.L. Blake and C.J. Merz. UCI Repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html (1998).

[84] Rave Harpaz and Robert Haralick. Fast relational matching. Technical Report TR-1401, ALPHATECH Inc., 2003.

[85] Rene Vidal. Generalized principal component analysis (GPCA): An algebraic geometric approach to subspace clustering and motion segmentation. Ph.D. Thesis, Electrical Engineering and Computer Sciences, University of California at Berkeley, 2003.

[86] Michael Tipping and Chris Bishop. Mixtures of principal component analyzers. Technical report NCRG/97/003, Neural Computing Research Group, Aston University, Aston Street, Birmingham, B4 7ET, UK., 1997.

[87] Ales Leonardis, Horst Bischof, and Jasna Maver. Multiple eigenspaces. *Pattern Recognition*, 35(11):2613–2627, 2002.

[88] Taku Yoshioka, Ryouko Morioka, Kazuo Kobayashi, Shigeyuki Oba, Naotake Ogawsawara, and Shin Ishii. Clustering of gene expression data by mixture

of PCA models. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 522–527, 2002.

[89] Geoffrey McLachlan, David Peel, and R. W. Bean. Modelling high-dimensional data by mixtures of factor analyzers. *Computational Statistics & Data Analysis*, 41(3-4):379–388, 2003.

[90] Geoffrey McLachlan. Mixtures of factor analyzers. In *Proceedings of the 17th International Conference on Machine Learning*, pages 599–606, 2000.

[91] R. A. Choudrey and S. J. Roberts. Variational mixture of Bayesian independent component analyzers. *Neural Computation*, 15(1):213–252, 2003.

[92] Jeffrey Ho, Ming-Hsuan Yang, Jongwoo Lim, Kuang-Chih Lee, and David J. Kriegman. Clustering appearances of objects under varying illumination conditions. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pages 11–18, 2003.

[93] David Edwards. *Introduction to Graphical Modelling*. Springer, 2000.

[94] Ron Larson, Bruce H. Edwards, and David C. Falvo. *Elementary Linear Algebra*. Houghton Mifflin Company, 2002.

[95] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[96] Saeed Tavazoie, Jason D. Hughes, Michael J. Campbell, Raymond J. Cho, and George M. Church. Systematic determination of genetic network architecture. *Nature Genetics*, 22(3):281 – 285, 1999.

[97] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine. Broad patterns of gene expression revealed by clustering analysis of

tumor and normal colon tissues probed by oligonucleotide arrays. In *Proceedings of the National Academy of Sciences*, volume 96, pages 6745–6750, 1999.

[98] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[99] J. Illingworth and J. Kittler. A survey of the hough transform. *Computer Vision, Graphics, and Image Processing*, 44(1):87–116, 1988.

[100] Isabelle Guyon and Andre Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[101] Dimitrios Gunopulos, Heikki Mannila, and Sanjeev Saluja. Discovering all most specific sentences by randomized algorithms. In *Proceedings of the 6th International Conference on Database Theory*, pages 215–229, 1997.

[102] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.