Clustering, Dimensionality Reduction, and Side Information

By

Hiu Chung Law

A Dissertation

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science & Engineering

2006

Abstract

Clustering, Dimensionality Reduction, and Side Information

By

Hiu Chung Law

Recent advances in sensing and storage technology have created many high-volume, high-dimensional data sets in pattern recognition, machine learning, and data mining. Unsupervised learning can provide generic tools for analyzing and summarizing these data sets when there is no well-defined notion of classes. The purpose of this thesis is to study some of the open problems in two main areas of unsupervised learning, namely clustering and (unsupervised) dimensionality reduction. Instance-level constraint on objects, an example of side-information, is also considered to improve the clustering results.

Our first contribution is a modification to the isometric feature mapping (ISOMAP) algorithm when the input data, instead of being all available simultaneously, arrive sequentially from a data stream. ISOMAP is representative of a class of nonlinear dimensionality reduction algorithms that are based on the notion of a manifold. Both the standard ISOMAP and the landmark version of ISOMAP are considered. Experimental results on synthetic data as well as real world images demonstrate that the modified algorithm can maintain an accurate low-dimensional representation of the data in an efficient manner.

We study the problem of feature selection in model-based clustering when the number of clusters is unknown. We propose the concept of feature saliency and introduce an expectation-maximization (EM) algorithm for its estimation. By using the minimum message length (MML) model selection criterion, the saliency of irrelevant features is driven towards zero, which corresponds to performing feature selection. The use of MML can also determine the number of clusters automatically by pruning away the weak clusters. The proposed algorithm is validated on both synthetic data and data sets from the UCI machine learning repository.

We have also developed a new algorithm for incorporating instance-level constraints in model-based clustering. Its main idea is that we require the cluster label of an object to be determined only by its feature vector and the cluster parameters. In particular, the constraints should not have any direct influence. This consideration leads to a new objective function that considers both the fit to the data and the satisfaction of the constraints simultaneously. The line-search Newton algorithm is used to find the cluster parameter vector that optimizes this objective function. This approach is extended to simultaneously perform feature extraction and clustering under constraints. Comparison of the proposed algorithm with competitive algorithms over eighteen data sets from different domains, including text categorization, low-level image segmentation, appearance-based vision, and benchmark data sets from the UCI machine learning repository, shows the superiority of the proposed approach.

To My Lord Jesus Christ

Acknowledgments

For the LORD gives wisdom,
and from his mouth come knowledge and understanding.

Proverbs 2:6

To our God and Father be glory for ever and ever. Amen.

Philippians 4:20

Time flies and I shall leave Michigan State soon – a place I shall cherish long after my graduation. There are so many people who have been so kind and so helpful to me during all these years; all of you have made a mark in my life!

First and foremost, I want to express my greatest gratitude to my thesis supervisor Dr. Anil Jain. He is such a wonderful advisor, mentor, and motivator. Under his guidance, I have learned a lot in different aspects of conducting research, including finding a good research problem, writing a convincing technical paper, and prioritizing different research tasks, to name a few. Of course I shall never forget all the good times when we "Prippies" partied in his awesome house. I am also very thankful to the rest of my thesis guidance committee, including Dr. John Weng, Dr. Bill Punch, and Dr. Sarat Dass. Their advice and suggestions have been very helpful.

I am also grateful to several other researchers who have mentored me during my various stages as a research student. Dr. Mário Figueiredo from Instituto Superior Técnico in Lisbon is a very nice person and his research insight has been an eye-opener. His intelligent use of the EM algorithm is truly remarkable. I feel fortunate that I have had the chance to work under the supervision of Dr. Paul Viola at Microsoft Research. Interaction with him has not only led to a much deeper appreciation of boosting, but also has sharpened my thoughts on how to formalize a research problem. It is also a great pleasure that I could work under Dr. Tin-Kam Ho at Bell Labs. Discussions with her have led to a new perspective towards different tools in pattern recognition and machine learning. The emphasis of Dr. Joachim Buhmann at ETH Zurich on correct modeling has impacted me on how to design a solution for any research problem. I am particularly grateful to Dr. Buhmann for his invitation to spend a month at ETH Zurich, and the hospitality he showed while I was there. The chance to work closely with Tilman Lange is definitely memorable. It is so interesting when two minds from different cultures and research heritage meet and conduct research together. Despite our differences, we have so much in common and the friendship with Tilman is probably the most valuable "side-product" of the research conducted during my Ph.D. study. ☺

I want to thank Dr. Yunhong Wang for providing the NLPR database that is used in chapters four and five in this thesis. The work in chapter five of this thesis has benefited from the discussions I had during my stay at ETH Zurich. Special thanks go to ONR (grant nos. # N00014-01-1-0266 and # N00014-04-1-0183) for its financial support during my Ph.D. studies.

On a more personal side, I am grateful to all the new friends that I have made during the past few years. I am especially grateful to the hospitality shown by the couples Steve & Liana, John (Ho) & Agnes, Ellen & her husband Mr. Yip, and John (Bankson) & Bonnie towards an international student like me. They have set up such a great example for me to imitate wherever I go. All the people in the Cantonese group of the Lansing Chinese Christian Ministry have given a "lab-bound" graduate student like me the possibility of a social life. The support shown by the group, including Simon, Paul, Kok, Tom, Timothy, Anthony, Twinsen, Dennis, Josie, Mitzi, Melody, Karen, Esther, Janni, Bean, Lok, Christal, Janice, and many more, has helped me to survive the tough times. All of my labmates in the PRIP lab, including Arun, Anoop, Umut, Xiaoguang, Hong, Dirk, Miguel, Yi, Unsang, Karthik, Pavan, Meltem, Sasha, Steve, . . . , have been so valuable to me. In addition to learning from them professionally, their emotional and social support is something I shall never forget.

Let me reserve my final appreciation to the most important people in my life. Without the nurturing, care, and love from my father and mother, I definitely could not have completed my doctoral degree. They have provided such a wonderful environment for me and my two brothers growing up. It is such a great achievement for my parents that all their three sons have completed at least a master's degree. I am also proud of my two brothers. I miss you, dad, mum, Pong, and Fai!

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

The most important characteristic of the information age is the abundance of data. Advances in computer technology, in particular the Internet, have led to what some people call "data explosion": the amount of data available to any person has increased so much that it is more than he or she can handle. According to a recent study[1] conducted at UC Berkeley, the amount of new data stored on paper, film, magnetic, and optical media is estimated to have grown 30% per year between 1999 and 2002. In the year 2002 alone, about 5 exabytes of new data have been generated. (One exabyte is about 10^{18} bytes, or 1000000 terabytes). Most of the original data are stored in electronic devices like hard disks (Table 1.1). This increase in both the volume and the variety of data calls for advances in methodology to understand, process, and summarize the data. From a more technical point of view, understanding the structure of large data sets arising from the data explosion is of fundamental importance in data mining, pattern recognition, and machine learning. In this thesis, we focus on two important techniques for data analysis in pattern recognition: dimensionality reduction and clustering. We also investigate how the addition of constraints, an example of side-information, can assist in data clustering.

1.1 Data Analysis

The word "data," as simple as it seems, is not easy to define precisely. We shall adopt a pattern recognition perspective and regard data as the description of a set of objects or patterns that can be processed by a computer. The objects are assumed to have some commonalities, so that the same systematic procedure can be applied to all the objects to generate the description.

1.1.1 Types of Data

Data can be classified into different types. Most often, an object is represented by the results of measurement of its various properties. A measurement result is called "a feature" in pattern recognition or "a variable" in statistics. The concatenation of all the features of a single object forms the feature vector. By arranging the feature vectors of different objects in different rows, we get a pattern matrix (also called "data matrix") of size n by d, where n is the total number of objects and d is the number of features. This representation is very popular because it converts different kinds of objects into a standard representation. If all the features are numerical, an object can be

[1] `http://www.sims.berkeley.edu/research/projects/how-much-info-2003/`

Table 1.1: Worldwide production of original data, if stored digitally, in terabytes (TB) circa 2002. Upper estimates (denoted by "upper") assume the data are digitally scanned, while lower estimates (denoted by "lower") assume the digital contents have been compressed. It is taken from Table 1.2 in `http://www.sims.berkeley.edu/research/projects/how-much-info-2003/execsum.htm`. The precise definitions of "paper," "film," "magnetic," and "optical" can be found in the web report.

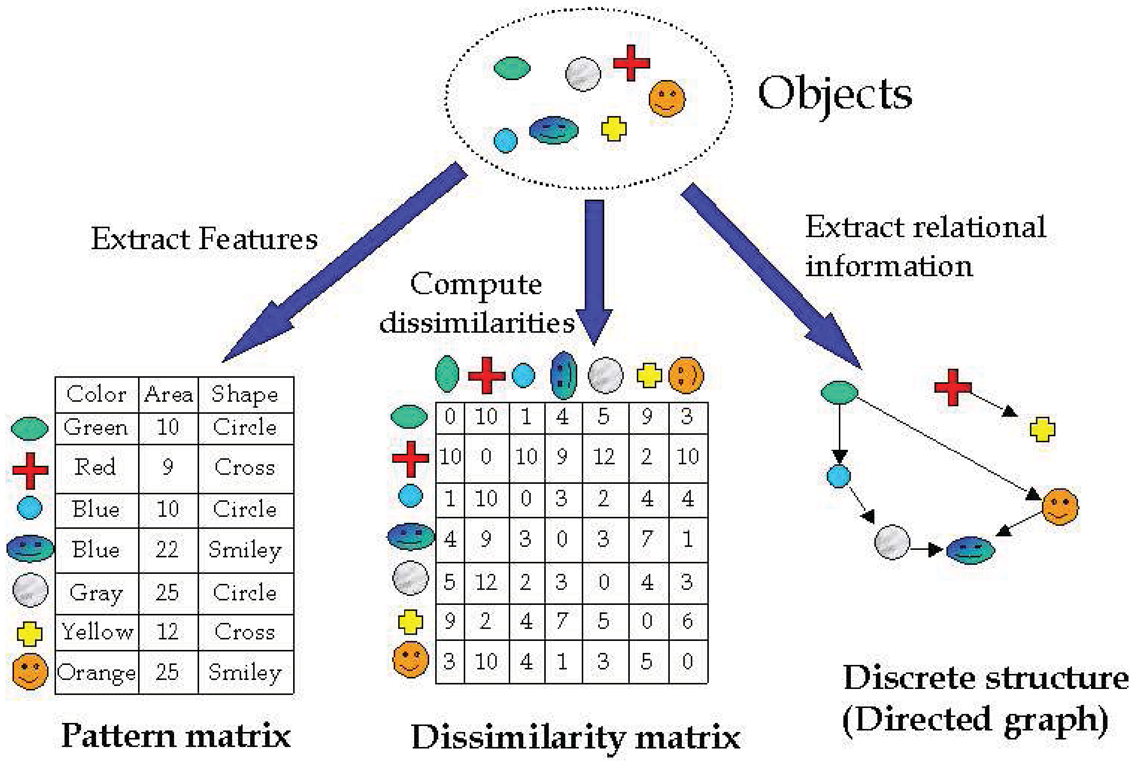
storage medium	upper, 2002	lower, 2002	upper, 1999–2000	lower, 1999-2000	% change, upper
Paper	1,634	327	1,200	240	36%
Film	420,254	74,202	431,690	58,209	-3%
Magnetic	5,187,130	3,416,230	2,779,760	2,073,760	87%
Optical	103	51	81	29	28%
Total	5,609,121	3,416,281	3,212,731	2,132,238	74.5%

represented as a point in $\mathbb{R}^d$. This enables a number of mathematical tools to be used to analyze the objects.

Alternatively, the similarity or dissimilarity between pairs of objects can be used as the data description. Specifically, a dissimilarity (similarity) matrix of size n by n can be formed for the n objects, where the (i, j)-th entry of the matrix corresponds to a quantitative assessment of how dissimilar (similar) the i-th and the j-th objects are. Dissimilarity representation is useful in applications where domain knowledge suggests a natural comparison function, such as the Hausdorff distance for geometric shapes. Examples of using dissimilarity for classification can be seen in [132], and more recently in [202]. Pattern matrix, on the other hand, can be easier to obtain than dissimilarity matrix. The system designer can simply list all the interesting attributes of the objects to obtain the pattern matrix, while a good dissimilarity measure with respect to the task can be difficult to design.

Similarity/dissimilarity matrix can be regarded as more generic than pattern matrix, because given the feature vectors of a set of objects, a dissimilarity matrix of these objects can be generated by computing the distances among the data points represented by these feature vectors. A similarity matrix can be generated either by subtracting the distances from a pre-specified number, or by exponentiating the negative of the distances. Pattern matrix, on the other hand, can be more flexible because the user can adjust the distance function according to the task. It is easier to incorporate new information by creating additional features than modifying the similarity/dissimiliarity measure. Also, in the common scenarios where there are a large number of patterns and a moderate number of features, the size of pattern matrix, $O(nd)$, is smaller than the size of similarity/dissimilarity matrix, $O(n^2)$.

A third possibility to represent an object is by discrete structures, such as parse trees, ranked lists, or general graphs. Objects such as chemical structures, web pages with hyperlinks, DNA sequences, computer programs, or customer preference for certain products have a natural discrete structure representation. Graph-related representations have also been used in various computer vision tasks, such as object recognition [145] and shape-from-shading [217]. Representing structural objects using a vector of attributes can discard important information on the relationship between different parts of the objects. On the other hand, coming up with the appropriate dissimilarity or similarity measure for such objects is often difficult. New algorithms that can handle discrete structure directly have been developed. An example is seen in [154], where a kernel function (diffusion kernel) is defined on different vertices in a graph, leading to improved classification performance for categorical data. Learning with structural data is sometimes called "learning with relational data,"



Color	Area	Shape
Green	10	Circle
Red	9	Cross
Blue	10	Circle
Blue	22	Smiley
Gray	25	Circle
Yellow	12	Cross
Orange	25	Smiley

0	10	1	4	5	9	3
10	0	10	9	12	2	10
1	10	0	3	2	4	4
4	9	3	0	3	7	1
5	12	2	3	0	4	3
9	2	4	7	5	0	6
3	10	4	1	3	5	0

Figure 1.1: Comparing feature vector, dissimilarity matrix and a discrete structure on a set of artificial objects. (Left) Extracting different features (color, area, and shape in this case) leads to a pattern matrix. (Center) A dissimilarity measure on the objects can be used to compare different pairs of objects, leading to a dissimilarity matrix. (Right) If the user can provide relational properties on the objects, a discrete structure like a directed graph can be created.

and several workshops[2] have been organized on this theme.

In Figure 1.1, we provide a simple illustration contrasting feature vector, dissimilarity matrix, and discrete structure representation for a set of artificial objects. Each of the representations corresponds to a different view of the objects. In practice, the system designer has to choose the representation that he or she thinks is the most relevant to the task.

In this thesis, we focus on feature vector representation, though dissimilarity/similarity information in the form of instance-level constraints is also considered.

1.1.2 Types of Features

Even within the feature vector representation, descriptions of an object can be classified into different types. A feature is essentially a measurement, and the "scale of measurement" [244] proposed by Stevens can be used to classify features into different categories. They are:

Nominal: discrete, unordered. Examples: "apple," "orange," and "banana."

Ordinal: discrete, ordered. Examples: "conservative," "moderate," and "liberal".

[2]A NIPS workshop in 2002 (`http://mlg.anu.edu.au/unrealdata/`) and several ICML workshops (2004:`http://www.cs.umd.edu/projects/srl2004/`) (2002:`http://demo.cs.brandeis.edu/icml02ws/`) (2000:`http://www.informatik.uni-freiburg.de/~ml/icml2000_workshop.html`) have been held on how to learn with structural or relational data.

Interval: continuous, no absolute zero, can be negative. Examples: temperature in Fahrenheit.

Ratio: continuous, with absolute zero, positive. Examples: length, weight.

This classification scheme, however, is not perfect [256]. One problem is that a measurement may not fit well into any of the categories listed in this scheme. An example for this is given in chapter 5 in [191], which considers the following types of measurements:

Grades: ordered labels such as Freshmen, Sophomore, Junior, Senior.

Ranks: starting from 1, which may be the largest or the smallest.

Counted fractions: bounded by zero and one. It includes percentage, for example.

Counts: non-negative integers.

Amounts: non-negative real numbers.

Balances: unbounded, positive, or negative values.

Most people would agree that these six types of data are different, yet all but the third and the last would be "ordinal" in the scheme by Stevens. "Counted fractions" also do not fit well into any of the category proposed by Stevens.

Consideration of different types of features can help us to design appropriate algorithms for handling different types of data arising from different domains.

1.1.3 Types of Analysis

The analysis to be performed on the data can also be classified into different types. It can be exploratory/descriptive, meaning that the investigator does not have a specific goal and only wants to understand the general characteristics or structure of the data. It can be confirmatory/inferential, meaning that the investigator wants to confirm the validity of a hypothesis/model or a set of assumptions using the available data. Many statistical techniques have been proposed to analyze the data, such as analysis of variance (ANOVA), linear regression, canonical correlation analysis (CCA), multidimensional scaling (MDS), factor analysis (FA), or principal component analysis (PCA), to name a few. A useful overview is given in [245].

In pattern recognition, most of the data analysis is concerned with predictive modeling: given some existing data ("training data"), we want to predict the behavior of the unseen data ("testing data"). This is often called "machine learning" or simply "learning." Depending on the type of feedback one can get in the learning process, three types of learning techniques have been suggested [68]. In supervised learning, labels on data points are available to indicate if the prediction is correct or not. In unsupervised learning, such label information is missing. In reinforcement learning, only the feedback after a sequence of actions that can change the possibly unknown state of the system is given. In the past few years, a hybrid learning scenario between supervised and unsupervised learning, known as semi-supervised learning, transductive learning [136], or learning with unlabeled data [195], has emerged, where only some of the data points have labels. This scenario happens frequently in applications, since data collection and feature extraction can often be automated, whereas the labeling of patterns or objects has to be done manually and this is expensive both in time and cost. In Chapter 5 we shall consider another hybrid scenario where instance-level constraints, which can be viewed as a "relaxed" version of labels, are available on some of the data points.

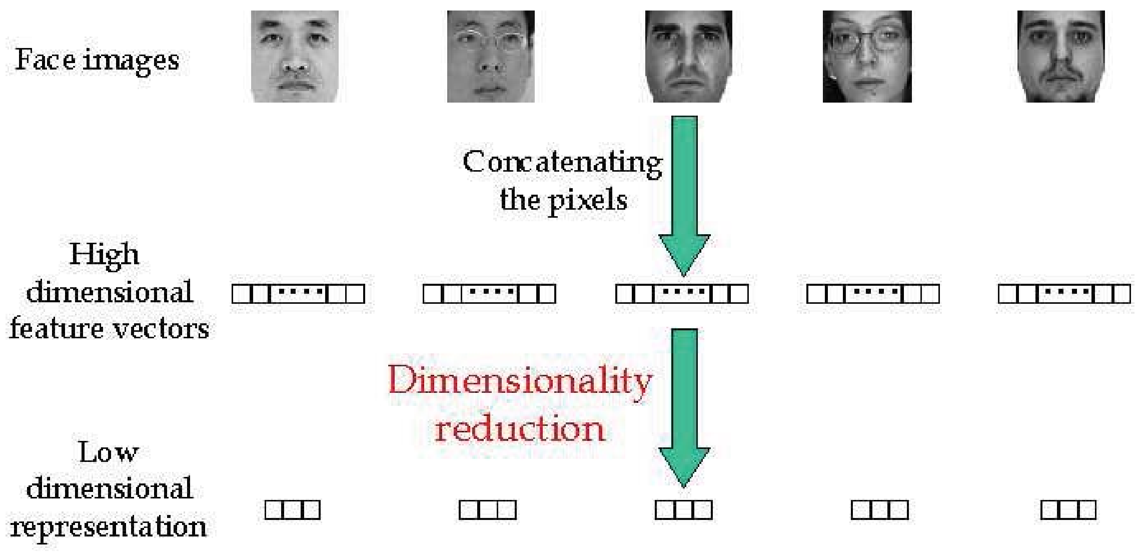

Face images
Concatenating the pixels
High dimensional feature vectors
Dimensionality reduction
Low dimensional representation

Figure 1.2: An example of dimensionality reduction. The face images are converted into a high dimensional feature vector by concatenating the pixels. Dimensionality reduction is then used to create a set of more manageable low-dimensional feature vectors, which can then be used as the input to various classifiers.

1.2 Dimensionality Reduction

Dimensionality reduction deals with the transformation of a high dimensional data set into a low dimensional space, while retaining most of the useful structure in the original data. An example application of dimensionality reduction with face images can be seen in Figure 1.2. Dimensionality reduction has become increasingly important due to the emergence of many data sets with a large number of features. The underlying assumption for dimensionality reduction is that the data points do not lie randomly in the high-dimensional space; rather, there is a certain structure in the locations of the data points that can be exploited, and the useful information in high dimensional data can be summarized by a small number of attributes.

1.2.1 Prevalence of High Dimensional Data

High dimensional data have become prevalent in different applications in pattern recognition, machine learning, and data mining. The definition of "high dimensional" has also changed from tens of features to hundreds or even tens of thousands of features [101].

Some recent applications involving high dimensional data sets include: (i) text categorization, the representation of a text document or a web page using the popular bag-of-words model can lead to thousands of features [277, 254], where each feature corresponds to the occurrence of a keyword or a key-term in the document; (ii) appearance-based computer vision approaches interpret each pixel as a feature [253, 22]. Images of handwritten digits can be recognized using the pixel values by neural networks [170] or support vector machines [255]. Even for a small image with size 64 by 64, such representation leads to more than 4,000 features; (iii) hyperspectral images[3] in remote sensing lead to high dimensional data sets: each pixel can contain more than 200 spectral

[3]Information on hyperspectral images can be found at `http://backserv.gsfc.nasa.gov/nips2003hyperspectral.html` and `http://www.eoc.csiro.au/hswww/Overview.htm`.

measurements in different wavelengths; (iv) the characteristics of a chemical compound recorded by a mass spectrometer can be represented by hundreds of features, where each feature corresponds to the reading in a particular range; (v) microarray technology enables us to measure the expression levels of thousands of genes simultaneously for different subjects with different treatments [6, 273]. Analyzing microarray data is particularly challenging, because the number of data points (subjects in this case) is much smaller than the number of features (expression levels in this case).

High dimensional data can also be derived in applications where the initial number of features is moderate. In an image processing task, the user can apply different filters with different parameters to extract a large number of features from a localized window in the image. The features are then summarized by applying a dimensionality reduction algorithm that matches the task at hand. This (relatively) automatic procedure contrasts with the traditional approach, where the user hand-crafts a small number of salient features manually, often with great effort. Creating a large feature set and then summarizing the features is advantageous when the domain is highly variable and robust features are hard to obtain, such as the occupant classification problem in [78].

1.2.2 Advantages of Dimensionality Reduction

Why should we reduce the dimensionality of a data set? In principle, the more information we have about each pattern, the better a learning algorithm is expected to perform. This seems to suggest that we should use as many features as possible for the task at hand. However, this is not the case in practice. Many learning algorithms perform poorly in a high dimensional space given a small number of learning samples. Often some features in the data set are just "noise" and thus do not contribute to (sometimes even degrade) the learning process. This difficulty in analyzing data sets with many features and a small number of samples is known as *the curse of dimensionality* [211].

Dimensionality reduction can circumvent this problem by reducing the number of features in the data set before the training process. This can also reduce the computation time, and the resulting classifiers take less space to store. Models with small number of variables are often easier for domain experts to interpret. Dimensionality reduction is also invaluable as a visualization tool, where the high dimensional data set is transformed into two or three dimensions for display purposes. This can give the system designer additional insight into the problem at hand.

The main drawback of dimensionality reduction is the possibility of information loss. When done poorly, dimensionality reduction can discard useful instead of irrelevant information. No matter what subsequent processing is to be performed, there is no way to recover this information loss.

1.2.2.1 Alternatives to Dimensionality Reduction

In the context of predictive modeling, (explicit) dimensionality reduction is not the only approach to handle high dimensional data. The naive Bayes classifier has found empirical success in classifying high dimensional data sets like webpages (the WEB$\rightarrow$KB project in [50]). Regularized classifiers such as support vector machines have achieved good accuracy for high dimensional data sets in the domain of text categorization [135]. Some learning algorithms have built-in feature selection abilities and thus (in theory) do not require explicit dimensionality reduction. For example, boosting [90] can use each feature as a "weak" classifier and construct an overall classifier by selecting the appropriate features and combining them [261].

Despite the apparent robustness of these learning algorithms in high dimensional data sets, it can still be beneficial to reduce the dimensionality first. Noisy features can degrade the performance of support vector machines because values of the kernel function (particular RBF kernel that depends on inter-point Euclidean distances) become less reliable if many features are irrelevant. It is beneficial to adjust the kernel to ignore those features [156], effectively performing dimensionality reduction. Concerns related to efficiency and storage requirement of a classifier also suggest the use of dimensionality reduction as a preprocessing step.

The important lesson is: dimensionality reduction is useful for most applications, yet the tolerance for the amount of information discarded should be subject to the judgement of the system designer. In general, a more conservative dimensionality reduction strategy should be employed if a classifier that is more robust to high dimensionality (such as support vector machines) is used. The dimensionality of the data may still be somewhat large, but at least little useful information is lost. On the other hand, if a more traditional and easier-to-understand classifier (like quadratic discriminant analysis) is to be used, we should reduce the dimensionality of the data set more aggressively to a smaller number, so that the classifier can competently handle the data.

1.2.3 Techniques for Dimensionality Reduction

Dimensionality reduction techniques can be broadly divided into several categories: (i) feature selection and feature weighting, (ii) feature extraction, and (iii) feature grouping.

1.2.3.1 Feature Selection and Feature Weighting

Feature selection, also known as variable selection or subset selection in the statistics (particularly regression) literature, deals with the selection of a subset of features that is most appropriate for the task at hand. A feature is either selected (because it is relevant) or discarded (because it is irrelevant). Feature weighting [271], on the other hand, assigns weights (usually between zero and one) to different features to indicate the saliencies of the individual features. Most of the literature on feature selection/weighting pertains to supervised learning (both classification [122, 151, 26, 101] and regression [186]).

Filters, Wrappers, and Embedded Algorithms Feature selection/weighting algorithms can be broadly divided into three categories [26, 151, 101]. The *filter* approaches evaluate the relevance of each feature (subset) using the data set alone, regardless of the subsequent learning task. RELIEF [147] and its enhancement [155] are representatives of this class, where the basic idea is to assign feature weights based on the consistency of the feature value in the k nearest neighbors of every data point. *Wrapper* algorithms, on the other hand, invoke the learning algorithm to evaluate the quality of each feature (subset). Specifically, a learning algorithm (*e.g.*, a nearest neighbor classifier, a decision tree, a naive Bayes method) is run using a feature subset and the feature subset is assessed by some estimate related to the classification accuracy. Often the learning algorithm is regarded as a "black box" in the sense that the wrapper algorithm operates independent of the internal mechanism of the classifier. An example is [212], which used genetic search to adjust the feature weights for the best performance of the k nearest neighbor classifier. In the third approach (called *embedded* in [101]), the learning algorithm is modified to have the ability to perform feature selection. There is no longer an explicit feature selection step; the algorithm automatically builds a classifier with a small number of features. LASSO (least absolute shrinkage and selection operator) [250] is a good example in this category. LASSO modifies the ordinary least square by including a constraint on the L_1 norm of the weight coefficients. This has the effect of preferring sparse regression coefficients (a formal statement for this is proved in [65, 64]), effectively performing feature selection. Another example is MARS (multivariate adaptive regression splines) [91], where choosing the variables used in the polynomial splines effectively performs variable selection. Automatic relevance detection in neural networks [177] is another example, which uses a Bayesian approach to estimate the weights in the neural network as well as the relevancy parameters that can be interpreted as feature weights.

Filter approaches are generally faster because they are classifier-independent and only require computation of simple quantities. They scale well with the number of features, and many of them can comfortably handle thousands of features. Wrapper approaches, on the other hand, can be superior in accuracy when compared with filters, which ignore the properties of the learning task at hand [151]. They are, however, computationally more demanding, and do not scale very well with the number of features. It is because training and evaluating a classifier with many features can be slow, and the performance of a traditional classifier with a large number of features may not be reliable enough to estimate the utilities of individual features. To get the best results from filters and wrappers, the user can apply a filter-type technique as preprocessing to cut down the feature set to a moderate size, and then use a wrapper algorithm to determine a small yet discriminative feature subset. Some state-of-the-art feature selection algorithms indeed adopt this approach, as observed in [102]. “Embedded” algorithms are highly specialized and it is difficult to compare them in general with filter and wrapper approaches.

Quality of a Feature Subset Feature selection/weighting algorithms can also be classified according to the definition of “relevance” or how the quality of a feature subset is assessed. Five definitions of relevance are given in [26]. Information-theoretic methods are often used to evaluate features, because the mutual information between a relevant feature and the class labels should be high [15]. Non-parametric methods can be used to estimate the probability density function of a continuous feature, which in turn is used to compute the mutual information [159, 251]. Correlation is also used frequently to evaluate features [278, 104]. A feature can be declared irrelevant if it is conditionally independent of the class labels given other features. The concept of Markov blanket is used to formalize this notion of irrelevancy in [153]. RELIEF [147, 155] uses the consistency of the feature value in the k nearest neighbors of every data point to quantify the usefulness of a feature.

Optimization Strategy Given a definition of feature relevancy, a feature selection algorithm can search for the most relevant feature subset. Because of the lack of monotonicity (with respect to the features) of many feature relevancy criteria, a combinatorial search through the space of all possible feature subsets is needed. Usually, heuristic (non-exhaustive) methods have to be adopted, because the size of this space is exponential in the number of features. In this case, one generally loses any guarantee of optimality of the selected feature subset. Different types of heuristics, such as sequential forward or backward searches, floating search, beam search, bi-directional search, and genetic search have been suggested [36, 151, 209, 275]. A comparison of some of these search heuristics can be found in [211]. In the context of linear regression, sequential forward search is often known as stepwise regression. Forward stagewise regression is a generalization of stepwise regression, where a feature is only “partially” selected by increasing the corresponding regression coefficient by a fixed amount. It is closely related to LASSO [250], and this relationship was established via least angle regression (LARS), another interesting algorithm on its own, in [72].

Wrapper algorithms generally include a heuristic search, as is the case for filter algorithms with feature quality criteria dependent on the features selected so far. Note that feature weighting algorithms do not involve a heuristic search because the weights for all features are computed simultaneously. However, the computation of the weights may be expensive. Embedded approaches also do not require any heuristic search. The optimal parameter is often estimated by optimizing a certain objective function. Depending on the form of the objective function, different optimization strategies can be used. In the case of LASSO, for example, a general quadratic programming solver, homotopy method [198], a modified version of LARS [72], or the EM algorithm [80] can be used to estimate the parameters.

1.2.3.2 Feature Extraction

In feature extraction, a small set of new features is constructed by a general mapping from the high dimensional data. The mapping often involves all the available features. Many techniques for feature extraction have been proposed. In this section, we describe some of the linear feature extraction methods, i.e., the extracted features can be written as linear combinations of the original features. Nonlinear feature extraction techniques are more sophisticated. In Chapter 2 we shall examine some of the recent nonlinear feature extraction algorithms in more detail. The readers may also find two recent surveys [284, 34] useful in this regard.

Unsupervised Techniques "Unsupervised" here refers to the fact that these feature extraction techniques are based only on the data (pattern matrix), without pattern label information. Principal component analysis (PCA), also known as Karhunen-Loeve Transform or simply KL transform, is arguably the most popular feature extraction method. PCA finds a hyperplane such that, upon projection to the hyperplane, the data variance is best preserved. The optimal hyperplane is spanned by the principal components, which are the leading eigenvectors of the sample covariance matrix. Features extracted by PCA consist of the projection of the data points to different principal components. When the features extracted by PCA are used for linear regression, it is sometimes called "principal component regression". Recently, sparse variants of PCA have also been proposed [137, 291, 52], where each principal component only has a small number of non-zero coefficients.

Factor analysis (FA) can also be used for feature extraction. FA assumes that the observed high dimensional data points are the results of a linear function (expressed by the factor loading matrix) on a few unobserved random variables, together with uncorrelated zero-mean noise. After estimating the factor loading matrix and the variance of the noise, the factor scores for different patterns can be estimated and serve as a low-dimensional representation of the data.

Supervised Techniques Labels in classification and response variables in regression can be used together with the data to extract more relevant features. Linear discriminant analysis (LDA) finds the projection direction such that the ratio of between-class variance to within-class variance is the largest. When there are more than two classes, multiple discriminant analysis (MDA) finds a sequence of projection directions that maximizes a similar criterion. Features are extracted by projecting the data points to these directions.

Partial least squares (PLS) can be viewed as the regression counterpart of LDA. Instead of extracting features by retaining maximum data variance as in principal component regression, PLS finds projection directions that can best explain the response variable. Canonical correlation analysis (CCA) is a closely related technique that finds projection directions that maximize the correlation between the response variables and the features extracted by projection.

1.2.3.3 Feature Grouping

In feature grouping, new features are constructed by combining several existing features. Feature grouping can be useful in scenarios where it can be more meaningful to combine features due to the characteristics of the domain. For example, in a text categorization task different words can have similar meanings and combining them into a single word class is more appropriate. Another example is the use of power spectrum for classification, where each feature corresponds to the energy in a certain frequency range. The preset boundaries of the frequency ranges can be sub-optimal, and the sum of features from adjacent frequency ranges can lead to a more meaningful feature by capturing the energy in a wider frequency range. For gene expression data, genes that are similar may share a common biological pathway and the grouping of predictive genes can be of interest to biologists [108, 230, 59].

The most direct way to perform feature grouping is to cluster the features (instead of the objects) of a data set. Feature clustering is not new; the SAS/STAT procedure "varclus" for variable clustering was written before 1990 [225]. It is performed by applying the hierarchical clustering method on a similarity matrix of different features, which is derived by, say, the Pearson's correlation coefficient. This scheme was probably first proposed in [124], which also suggested summarizing one group of features by a single feature in order to achieve dimensionality reduction. Recently, feature clustering has been applied to boost the performance in text categorization. Techniques based on distribution clustering [4], mutual information [62], and information bottleneck [238] have also been proposed.

Features can also be clustered together with the objects. As mentioned in [201], this idea has been known under different names in the literature, including "bi-clustering" [41, 150], "co-clustering" [63, 61], "double-clustering" [73], "coupled clustering" [95], and "simultaneous clustering" [208]. A bipartite graph can be used to represent the relationship between objects and features, and the partitioning of the graph can be used to cluster the objects and the features simultaneously [281, 61]. Information bottleneck can also be used for this task [237].

In the context of regression, feature grouping can be achieved indirectly by favoring similar features to have similar coefficients. This can be done by combining ridge regression with LASSO, leading to the elastic net regression algorithm [290].

1.3 Data Clustering

The goal of (data) clustering, also known as cluster analysis, is to discover the "natural" grouping(s) of a set of patterns, points, or objects. Webster[4] defines cluster analysis as "a statistical classification technique for discovering whether the individuals of a population fall into different groups by making quantitative comparisons of multiple characteristics." An example of clustering can be seen in Figure 1.3. The unlabeled data set in Figure 1.3(a) is assigned labels by a clustering procedure in order to discover the natural grouping of the three groups as shown in Figure 1.3(b).

Cluster analysis is prevalent in any discipline that involves analysis of multivariate data. It is difficult to exhaustively list the numerous uses of clustering techniques. Image segmentation, an important problem in computer vision, can be formulated as a clustering problem [94, 128, 234]. Documents can be clustered [120] to generate topical hierarchies for information access [221] or retrieval [20]. Clustering is also used to perform market segmentation [3, 39] as well as to study genome data [6] in biology.

Clustering, unfortunately, is difficult for most data sets. A non-trivial example of clustering is shown in Figure 1.4. Unlike the three well-separated, spherical clusters in Figure 1.3, the seven clusters in Figure 1.4 have diverse shapes: globular, circular, and spiral in this case. The densities and the sizes of the clusters are also different. The presence of background noise makes the detection of the clusters even more difficult. This example also illustrates the fundamental difficulty of clustering. The diversity of "good" clusters in different scenarios make it virtually impossible for one to provide a universal definition of "good" clusters. In fact, it has been proved in [149] that it is impossible for any clustering algorithm to achieve some fairly basic goals simultaneously. Therefore, it is not

[4]`http://www.m-w.com/`

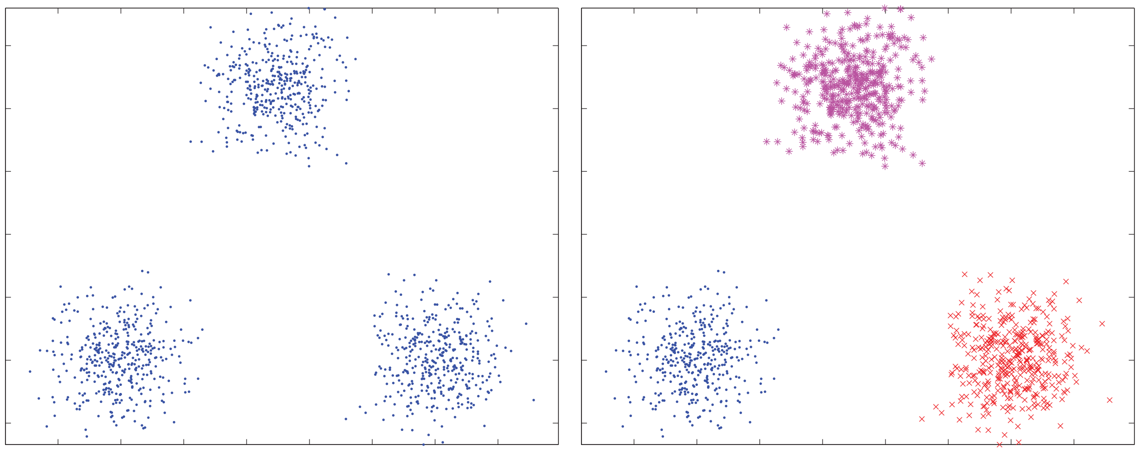

(a) Original data (b) Clustering Result

Figure 1.3: The three well-separated clusters can be easily detected by most clustering algorithms. **Images in this thesis/dissertation are presented in color.**

surprising that many clustering algorithms have been proposed to address the different needs of "good clusters" in different scenarios.

In this section, we attempt to provide a taxonomy of the major clustering techniques, present a brief history of cluster analysis, and present the basic ideas of some popular clustering algorithms in the pattern recognition community.

1.3.1 A Taxonomy of clustering

Many clustering algorithms have been proposed in different application scenarios. Perhaps the most important way to classify clustering algorithms is *hierarchical* versus *partitional*. Hierarchical clustering creates a tree of objects, where branches merging at the lower levels correspond to higher similarity. Partitional clustering, on the other hand, aims at creating a "flat" partition of the set of objects with each object belonging to one and only one group.

Clustering algorithms can also be classified by the type of input data used (pattern matrix or similarity matrix), or by the type of the features, e.g. numerical, categorical, or special data structures, such as rank data, strings, graphs, etc. (See Section 1.1.1 for information on different types of data.) Alternatively, a clustering algorithm can be characterized by the probability model used, if any, or by the core search (optimization) process used to find the clusters. Hierarchical clustering algorithms can be described by the clustering direction, either agglomerative or divisive.

In Figure 1.5, we provide one possible hierarchy of partitional clustering algorithms (modified from [131]). Heuristic-based techniques refer to clustering algorithms that optimize a certain notion of "good" clusters. The goodness function is constructed by the user in a heuristic manner. Model-based clustering assumes that there are underlying (usually probabilistic) models that govern the clusters. Density-based algorithms attempt to estimate the data density and utilize that to construct the clusters.

One may further sub-divide heuristic-based techniques depending on the input type. If a pattern matrix is used, the algorithm is usually prototype-based, i.e., each cluster is represented by the most typical "prototype." The k-means and the k-medoids algorithms [79] are probably the best known in this category. If a dissimilarity or similarity matrix is used as the input, two sub-categories are possible: those based on linkage (single-link, average-link, complete-link, and CHAMELEON [142]),

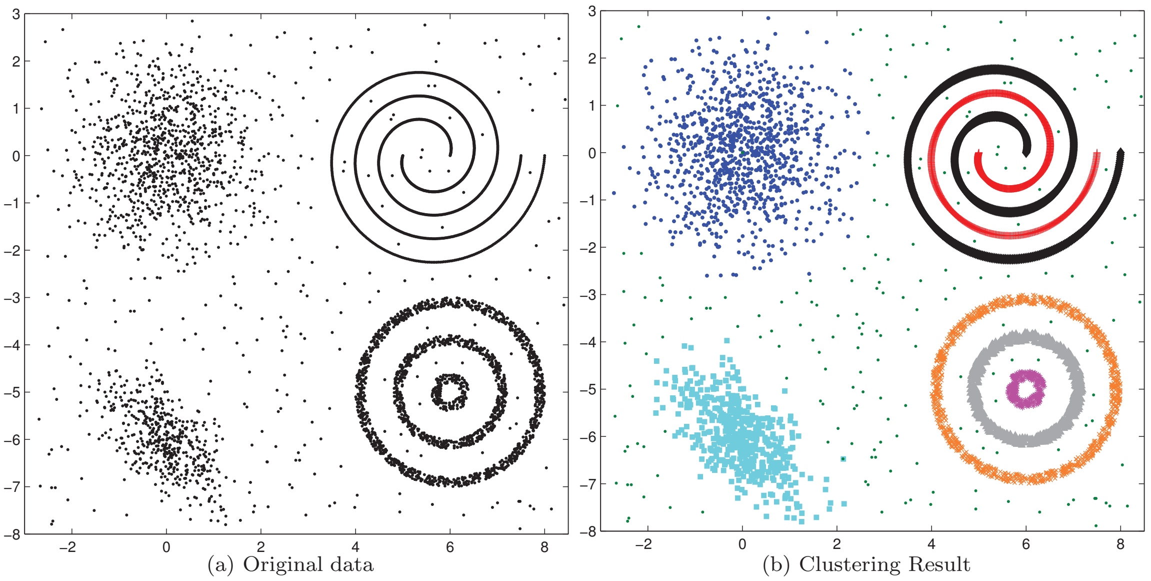


Figure 1.4: Diversity of clusters. The seven clusters in this data set (denoted by the seven different colors), though easily identified by human, are difficult to detect automatically. The clusters are of different shapes, sizes, and densities. The presence of background noise makes the clustering task even more difficult.

and those inspired from graph theory, such as min-cut [272] and spectral clustering [234, 194]. Model-based algorithms often refer to clustering by using a finite mixture distribution [184], with each mixture component interpreted as a cluster. Spatial clustering can involve a probabilistic model of the point process. For density-based methods, the mean-shift algorithm [45] finds the modes of the data densities by the mean-shift operation, and the cluster label is determined by which "basin of convergence" a point is located. DENCLUE [111] utilizes a kernel (non-parametric) estimate for the data density to find the clusters.

1.3.2 A Brief History of Cluster Analysis

According to the scholarly journal archive JSTOR[5], the first appearance of the word "cluster" in the title of a scholarly article was in 1739 [11]: "A Letter from John Bartram, M. D. to Peter Collinson, F. R. S. concerning a Cluster of Small Teeth Observed by Him at the Root of Each Fang or Great Tooth in the Head of a Rattle-Snake, upon Dissecting It". The word "cluster" here, though, was used only in its general sense to denote a group. The phrase "cluster analysis" first appeared in 1954 and it was suggested as a tool to understand anthropological data [43]. In its early days, cluster analysis was sometimes referred to as grouping [48, 85], and biologists called it "numerical taxonomy" [242].

Early research on hierarchical clustering was mainly done by biologists, because these techniques helped them to create a hierarchy of different species for analyzing their relationship systematically. According to [242], single-link clustering [240], complete-link clustering [213], and average-link clustering [241] first appeared in 1957, 1948, and 1958, respectively. Ward's method [266] was proposed in 1963. Partitional clustering, on the other hand, is closely related to data compression and vector quantization. This link is not surprising because the cluster labels assigned by a partitional cluster-

[5] `http://www.jstor.org`

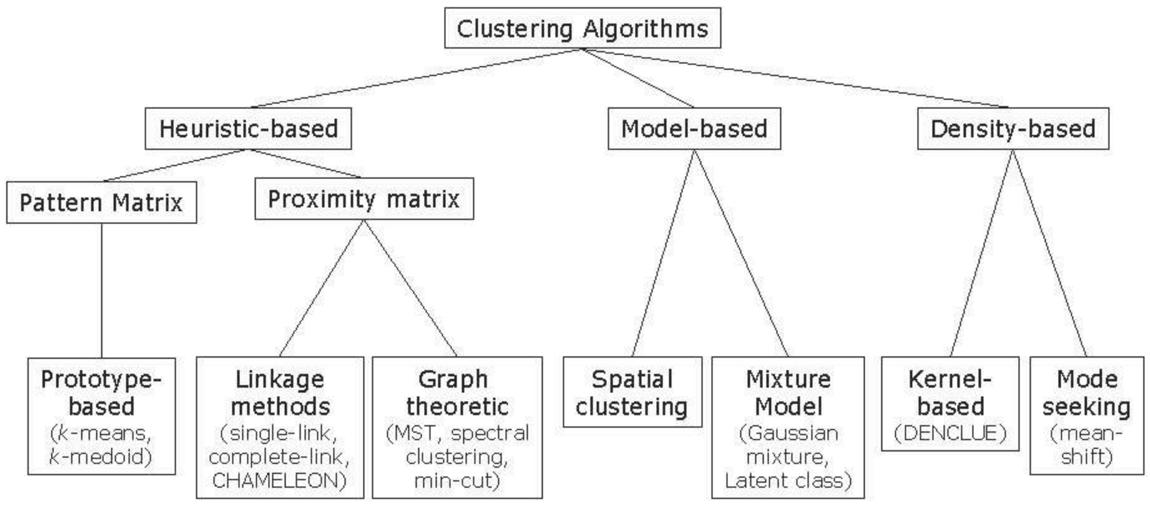

Clustering Algorithms
Heuristic-based
Model-based
Density-based
Pattern Matrix
Proximity matrix
Prototype-based (k-means, k-medoid)
Linkage methods (single-link, complete-link, CHAMELEON)
Graph theoretic (MST, spectral clustering, min-cut)
Spatial clustering
Mixture Model (Gaussian mixture, Latent class)
Kernel-based (DENCLUE)
Mode seeking (mean-shift)

Figure 1.5: A taxonomy of clustering algorithms.

ing algorithm can be viewed as the compressed version of the data. The most popular partitional clustering algorithm, k-means, has been proposed several times in the literature: Steinhaus in 1955 [243], Lloyd in 1957 [174], and MacQueen in 1967 [178]. The ISODATA algorithm by Ball and Hall in 1965 [8] can be regarded as an adaptive version of k-means that adjusts the number of clusters. The k-means algorithm is also attributed to Forgy (like [140] and [99]), though the reference for this [88] only contains an abstract and it is not clear what Forgy exactly proposed. The historical account of vector quantization given in [99] also presents the history of some of the partitional clustering algorithms. In 1971, Zahn proposed a graph-theoretic clustering method [280], which is closely related to single-link clustering. The EM algorithm, which is the standard algorithm for estimating a finite mixture model for mixture-based clustering, is attributed to Dempster *et al.* in 1977 [58]. Interest in mean-shift clustering was revived in 1995 by Cheng [40], and Comaniciu and Meer further popularized it in [45]. Hoffman and Buhmann considered the use of deterministic annealing for pairwise clustering [115], and Fischer and Buhmann modified the connectedness idea in single-link clustering that led to path-based clustering [84]. The normalized cut algorithm by Shi and Malik [233] in 1997 is often regarded as the first spectral clustering algorithm, though similar ideas were considered by spectral graph theorists earlier. A summary of the important results in spectral graph theory can be found in the 1997 book by Chung [42]. The emergence of data mining leads to a new line of clustering research that emphasizes efficiency when dealing with huge database. DBSCAN by Ester *et al.* [77] for density-based clustering and CLIQUE by Agrawal *et al.* [2] for subspace clustering are two well-known algorithms in this community.

The current literature on cluster analysis is vast, and hundreds of clustering algorithms have been proposed in the literature. It will require a tremendous effort to list and summarize all the major clustering algorithms. The reader is encouraged to refer to a survey like [130] or [79] for an overview of different clustering algorithms.

1.3.3 Examining Some Clustering Algorithms

In this section, we will examine two very important clustering algorithms used in the pattern recognition community: the k-means algorithm and the EM algorithm. Other clustering algorithms that are used regularly in pattern recognition include the mean-shift algorithm [45, 44, 40], pairwise clustering [115, 116], path-based clustering [84, 83], and spectral clustering [234, 139, 269, 194, 258, 42].

Let $\{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ be the set of n d-dimensional data points to be clustered. The cluster label of $\mathbf{y}_i$ is denoted by z_i. The goal of (partitional) clustering is to recover z_i, with $z_i \in \{1, \ldots, k\}$, where k denotes the number of clusters specified by the user. The set of $\mathbf{y}_i$ with $z_i = j$ is referred to as the j-th cluster.

1.3.3.1 The k-means algorithm

The k-means algorithm is probably the best known clustering algorithm. In this algorithm, the j-th cluster is represented by the "cluster prototype" $\boldsymbol{\mu}_j$ in $\mathbb{R}^d$. Clustering is done by finding z_i and $\boldsymbol{\mu}_j$ that minimize the following cost function:

$$J_{k-\text{means}} = \sum_{i=1}^{n} ||\mathbf{y}_i - \boldsymbol{\mu}_{z_i}||^2 = \sum_{i=1}^{n} \sum_{j=1}^{k} I(z_i = j) ||\mathbf{y}_i - \boldsymbol{\mu}_j||^2. \tag{1.1}$$

Here, $I(z_i = j)$ denotes the indicator function, which is one if the condition $z_i = j$ is true, and zero otherwise. To optimize $J_{k-\text{means}}$, we first assume that all $\boldsymbol{\mu}_j$ are specified. The values of z_i that minimize $J_{k-\text{means}}$ are given by

$$z_i = \arg\min_j ||\mathbf{y}_i - \boldsymbol{\mu}_j||^2. \tag{1.2}$$

On the other hand, if z_i is fixed, the optimal $\boldsymbol{\mu}_j$ can be found by differentiating $J_{k-\text{means}}$ with respect to $\boldsymbol{\mu}_j$ and setting the derivatives to zero, leading to

$$\boldsymbol{\mu}_j = \frac{\sum_{j=1}^{k} I(z_i = j) \boldsymbol{\mu}_j}{\sum_{j=1}^{k} I(z_i = j)} = \frac{\sum_{i=1, z_i=j}^{n} \boldsymbol{\mu}_j}{\text{number of } i \text{ with } z_i = j}. \tag{1.3}$$

Starting from an initial guess on $\boldsymbol{\mu}_j$, the k-means algorithm iterates between Equations (1.2) and (1.3), which is guaranteed to decrease the k-means objective function until a local minimum is reached. In this case, $\boldsymbol{\mu}_j$ and z_i remain unchanged after the iteration, and the k-means algorithm is said to have converged. The resulting z_i and $\boldsymbol{\mu}_j$ constitute the clustering solution. In practice, one can stop if the change in successive values of $J_{k-\text{means}}$ is less than a threshold.

The k-means algorithm is easy to understand and is also easy to implement. However, k-means has problems in discovering clusters that are not spherical in shape. It also encounters some difficulties when different clusters have a significantly different number of points. k-means also requires a good initialization to avoid getting trapped in a poor local minimum. In many cases, the user does not know the number of clusters in advance, which is required by k-means. The problem of determining the value of k automatically still does not have a very satisfactory solution. Some heuristics have been described in [125], and a recent paper on this is [106].

Because the k-means algorithm alternates between the two conditions of optimality, it is an example of alternating optimization. The k-means clustering result can be interpreted as a solution to vector quantization, with a codebook of size k and a square error loss function. Each $\boldsymbol{\mu}_j$ is a codeword in this case. The k-means algorithm can also be viewed as a special case of fitting a Gaussian mixture, with covariance matrices of all the mixture components fixed to be $\sigma^2\mathbf{I}$ and σ tends to zero (for the "hard" cluster assignment). The k-medoid algorithm is similar to k-means, except that $\boldsymbol{\mu}_j$ is restricted to be one of the given patterns $\mathbf{y}_i$.

There is also an online version of k-means. When the i-th data point $\mathbf{y}_i$ is observed, the cluster center $\boldsymbol{\mu}_j$ that is the nearest to $\mathbf{y}_i$ is found. $\boldsymbol{\mu}_j$ is then updated by

$$\boldsymbol{\mu}_j^{\text{new}} = \boldsymbol{\mu}_j + \alpha(\mathbf{y}_i - \boldsymbol{\mu}_j), \tag{1.4}$$

where α is the learning rate. This learning rule is an example of "winner-take-all" in competitive learning, because only the cluster that "wins" the data point can learn from it.

1.3.3.2 Clustering by Fitting Finite Mixture Model

The k-means algorithm is an example of "hard" clustering, where a data point is assigned to only one cluster. In many cases, it is beneficial to consider "soft" clustering, where a point is assigned to different clusters with different degrees of certainties. This can be done either by fuzzy clustering or by mixture-based clustering. We prefer the latter because it has a more rigorous foundation.

In mixture-based clustering, a finite mixture model is fitted to the data. Let Y and Z be the random variables for a data point and a cluster label, respectively. Each cluster is represented by the component distribution $p(Y|\theta_j)$, where θ_j denotes the parameter for the j-th cluster. Data points from the j-th cluster are assumed to follow this distribution, i.e., $p(Y|Z = j) = p(Y|\theta_j)$. The component distribution $p(Y|\theta_j)$ is often assumed to be a Gaussian when Y is continuous, and the corresponding mixture model is called "a mixture of Gaussians". If Y is categorical, multinomial distribution can be used for $p(Y|\theta_j)$. Let $\alpha_j = P(Z = j)$ be the prior probability for the j-th cluster. The key idea of a mixture model is

$$p(Y|\Theta) = \sum_{j=1}^{k} P(Y|Z = j)P(Z = j) = \sum_{j=1}^{k} \alpha_j p(Y|\theta_j), \tag{1.5}$$

where $\Theta = \{\theta_1, \ldots, \theta_k, \alpha_1, \ldots, \alpha_k\}$ contains all the model parameters. The mixture model can be understood as a two-stage data generation process. First, the hidden cluster label Z is sampled from a multinomial distribution with parameters $(\alpha_1, \ldots, \alpha_k)$. The data point Y is then generated according to the mixture distribution determined by Z, i.e., Y is sampled from $p(Y|\theta_j)$ if $Z = j$.

The degree of membership of $\mathbf{y}_i$ to the j-th cluster is determined by the posterior probability of Z equals to j given $\mathbf{y}_i$, i.e.,

$$p(Z = j|Y = \mathbf{y}_i) = \frac{p(Z = j, Y = \mathbf{y}_i)}{p(Y = \mathbf{y}_i)} = \frac{\alpha_j p(Y|\theta_j)}{\sum_{j=1}^{k} \alpha_j p(Y|\theta_j)}. \tag{1.6}$$

If a "hard" clustering is needed, $\mathbf{y}_i$ can be assigned to the cluster with the highest posterior probability $P(Z|Y = \mathbf{y}_i)$.

The parameter Θ can be determined using the maximum likelihood principle. We seek Θ that minimizes the negative log-likelihood:

$$J_{\text{mixture}} = -\sum_{i=1}^{n} \log \sum_{j=1}^{k} \alpha_j p(\mathbf{y}_i|\theta_j). \tag{1.7}$$

For brevity of notation, we write $p(\mathbf{y}_i|\theta_j)$ to denote $p(Y = \mathbf{y}_i|\theta_j)$.

The EM algorithm can be used to optimize J_{mixture}. EM is a powerful technique for parameter estimation when some of the data are missing. In the context of a finite mixture model, the missing data are the cluster labels. Starting with an initial guess of the parameters, the EM algorithm alternates between the "E-step" and the "M-step". Let $r_{ij} = P(Z = j|Y = \mathbf{y}_i, \Theta^{\text{old}})$, where

Θ^{old} is the current parameter estimate. In the E-step, we compute the expected complete data log-likelihood, also known as the Q-function:

$$Q(\Theta||\Theta^{\text{old}}) = E\left[\sum_{i=1}^{n} \log p(\mathbf{y}_i, z_i)\right] = \sum_{i=1}^{n}\sum_{j=1}^{k} r_{ij}\left(\log \alpha_j + \log p(\mathbf{y}_i|\theta_j)\right) \tag{1.8}$$

Note that the expectation is done with respect to the old parameter value via r_{ij}. Computationally, E-step requires calculation of r_{ij}. In the M-step, Θ that maximizes $Q(\Theta||\Theta^{\text{old}})$ is found:

$$\Theta^{\text{new}} = \arg\max_{\Theta} Q(\Theta||\Theta^{\text{old}}). \tag{1.9}$$

The M-step is guaranteed to decrease J_{mixture}. By repeating the E-step and the M-step, the negative log-likelihood continues to decrease until a local minimum is reached.

Convergence Proofs on the EM algorithm In this section, we shall state the well-known proof in the literature that the M-step indeed decreases J_{mixture}, thereby showing that the EM algorithm does converge to a local minimum of J_{mixture}. We consider the correctness of the EM algorithm in a more general setting, where Y and Z are redefined to mean "observed data" and "missing data," respectively. Note that the data points and the missing labels are examples of observed data and missing data, respectively.

In this general setting, $Q(\Theta||\Theta^{\text{old}})$ can be written as

$$Q(\Theta||\Theta^{\text{old}}) = \sum_{Z} p(Z|Y,\Theta^{\text{old}}) \log p(Y,Z|\Theta) \tag{1.10}$$

Our first proof is based on the concavity of the logarithm function. Because M-step maximizes $Q(\Theta)$, $Q(\Theta^{\text{new}}) - Q(\Theta^{\text{old}}) \geq 0$. Observe that

$$\begin{aligned}
& Q(\Theta^{\text{new}}) - Q(\Theta^{\text{old}}) \\
&= \sum_{Z} p(Z|Y,\Theta^{\text{old}})\left(\log p(Y,Z|\Theta^{\text{new}}) - \log p(Y,Z|\Theta^{\text{old}})\right) \\
&= \log p(Y|\Theta^{\text{new}}) - \log p(Y|\Theta^{\text{old}}) + \sum_{Z} p(Z|Y,\Theta^{\text{old}}) \log \frac{p(Z|Y,\Theta^{\text{new}})}{p(Z|Y,\Theta^{\text{old}})} \\
&\leq \log p(Y|\Theta^{\text{new}}) - \log p(Y|\Theta^{\text{old}}) + \log \sum_{Z} p(Z|Y,\Theta^{\text{old}}) \frac{p(Z|Y,\Theta^{\text{new}})}{p(Z|Y,\Theta^{\text{old}})} \\
&= \log p(Y|\Theta^{\text{new}}) - \log p(Y|\Theta^{\text{old}}).
\end{aligned}$$

The inequality is due to the concavity of logarithm, and the fact that $p(Z|Y,\Theta^{\text{old}})$ can be viewed as "weights" because they are non-negative and $\sum_Z p(Z|Y,\Theta^{\text{old}}) = 1$. Since $Q(\Theta^{\text{new}}) - Q(\Theta^{\text{old}}) \geq 0$, the above implies $\log p(Y|\Theta^{\text{new}}) - \log p(Y|\Theta^{\text{old}}) \geq 0$. So, the update of parameter from Θ^{old} to Θ^{new} indeed improves the log-likelihood of the observed data. When $\Theta^{\text{old}} = \Theta^{\text{new}}$, the inequality becomes an equality, and we reach a local minimum of $\log p(Y|\Theta)$.

Note that the above argument holds as long as $Q(\Theta^{\text{new}}) - Q(\Theta^{\text{old}}) \geq 0$. Thus it suffices to increase (instead of maximizes) the expected complete log-likelihood in the M-step. The resulting algorithm that only increases the expected complete log-likelihood is known as the generalized EM algorithm.

It is interesting to note a variant of the EM algorithm used in [80] for Bayesian parameter estimation. The goal is to find Θ that maximizes $\log p(\Theta|Y)$. Since the missing data in [80] are continuous, the expectation is performed by integration instead of summation. The E-step computes $\int p(\Theta|Z, Y) \log p(Z|\Theta^{\text{old}}, Y)\ dZ$, and the M-step solves $\Theta^{\text{new}} = \arg\max_\Theta \int p(Z|\Theta^{\text{old}}, Y) \log p(\Theta|Z, Y)\ dZ$. The correctness of this variant of the EM algorithm can be seen by the following:

$$
\begin{aligned}
&\int p(Z|\Theta^{\text{old}}, Y) \log p(\Theta^{\text{new}}|Z, Y)\ dZ - \int p(Z|\Theta^{\text{old}}, Y) \log p(\Theta^{\text{old}}|Z, Y)\ dZ \\
&= \int p(Z|\Theta^{\text{old}}, Y)\big(\log p(\Theta^{\text{new}}|Y) + \log p(Z|\Theta^{\text{new}}, Y) - \log P(Z|Y) \\
&\qquad - \log p(\Theta^{\text{old}}|Y) - \log p(Z|\Theta^{\text{old}}, Y) + \log P(Z|Y)\big)\ dZ \\
&= \log p(\Theta^{\text{new}}|Y) - \log p(\Theta^{\text{old}}|Y) + \int p(Z|\Theta^{\text{old}}, Y) \log \frac{p(Z|\Theta^{\text{new}}, Y)}{p(Z|\Theta^{\text{old}}, Y)}\ dZ \\
&\leq \log p(\Theta^{\text{new}}|Y) - \log p(\Theta^{\text{old}}|Y)
\end{aligned}
$$

Note that $p(\Theta|Z, Y) = p(\Theta|Y)p(Z|\Theta, Y)/p(Z|Y)$.

Our second proof of the EM algorithm is to regard it as a special case of variational method. Here, we follow the presentation in [205]. Let $T(Z)$ be an unknown variable distribution on the missing data Z. Since $p(Y|\Theta) = p(Y, Z|\Theta)/p(Z|Y, \Theta)$, we have

$$
\begin{aligned}
\log p(Y|\Theta) &= \log p(Y, Z|\Theta) - \log p(Z|Y, \Theta) \\
\log p(Y|\Theta) &= \sum_Z T(Z) \log p(Y, Z|\Theta) - \sum_Z T(Z) \log p(Z|Y, \Theta) \\
&= \sum_Z T(Z) \log \frac{p(Y, Z|\Theta)}{T(Z)} + D_{KL}(T(Z)||p(Z|Y, \Theta))
\end{aligned}
$$

Here, $D_{KL}(T(Z)||p(Z|Y))$ is the Kullback Leibler divergence defined as

$$
D_{KL}(TQ(Z)||p(Z|Y)) = \sum_Z TQ(Z) \log \frac{T(Z)}{p(Z|Y)}.
$$

Note that the divergence is always nonnegative, meaning that $s = \sum_Z T(Z) \log \frac{p(Y,Z|\Theta)}{T(Z)}$ is a lower bound of $\log p(Y|\Theta)$. Variational method maximizes $\log p(Y|\Theta)$ indirectly by finding Θ and $T(Z)$ that maximizes s, under a restriction on the form of $T(Z)$. The EM algorithm can be regarded as a special case of variational method, which does not put any restriction on $T(Z)$. It is easy to show that in this case s is maximized with respect to $T(Z)$ if $T(Z) = p(Z|Y, \Theta)$. With this choice of $T(Z)$, s is no longer a lower bound but exactly equals $\log p(Y|\Theta)$, because the divergence term is zero. Maximizing s with respect to Θ is the same as maximizing $\sum_Z p(Z|Y, \Theta) \log p(Y, Z|\Theta)$, which is the Q-function.

1.4 Side-Information

In many pattern recognition problems, the performance of advanced classifiers like support vector machines and simple classifiers like k-nearest neighbors are more or less the same. It is the "quality" of the input information (in terms of discrimination power), instead of the type of the classifier, that is the determining factor in the classification accuracy. However, research effort in pattern recognition and machine learning has focused on devising better classifiers. One is more likely to improve the performance of practical systems by incorporating additional domain/contextual information, than by improving the classifier. *Side-information*, i.e., information other than what is contained in feature vectors and class labels, is relevant here because it provides alternative means for the system designer to input more prior knowledge into the classficiation/clustering system, therefore boosting its performance.

Side-information arises because some aspects of a pattern recognition problem cannot be specified via the class labels and the feature vectors. It can be viewed as a complement to the given pattern or proximity matrix. Examples of side-information include alternative metrics between objects, known data groupings or associations, additional labels or attributes (such as soft biometric traits [123]), relevance of different features, and ranks of the objects.

Side-information is particularly valuable to clustering, owing to the inherent arbitrariness in the notion of a cluster. Given different possibilities to cluster a data set, side information can help us to identify the cluster structure that is the most appropriate in the context that the clustering solution will be used. A set of *constraints*, which specify the relationship between different cluster labels, is probably the most natural type of side-information in clustering. Constraints arise naturally in many clustering applications. For example, in image segmentation one can have partial grouping cues for several regions in the image to assist in the overall clustering [279]. Clustering of customers in a market-basket database can have multiple records pertaining to the same person. In video retrieval tasks different users may provide alternative annotations of images in small subsets of a large database [110]. Such groupings may be used for semi-supervised clustering of the entire database. "Orthogonality" to a known or trivial partition of the data set is another type of side-information for clustering, and this requirement can be incorporated via a variant of information bottleneck [97].

1.5 Overview

In the remainder of this thesis, we shall first provide an in-depth survey of some nonlinear dimensionality reduction methods in Chapter 2. We then present our work on how to convert ISOMAP, one of the algorithms described in Chapter 2, to its incremental version in Chapter 3. In Chapter 4, we present our algorithm on the problem of estimating the relevance of different features in a clustering context. Chapter 5 describes our proposed approach to perform model-based clustering in the presence of constraints. Finally, we conclude with some of our contributions to the field and outline some research directions in Chapter 6.

Chapter 2

A Survey of Nonlinear Dimensionality Reduction Algorithms

In section 1.2 we described the importance of dimensionality reduction and presented an overall picture of different approaches for dimensionality reduction. This chapter continues the discussion in section 1.2.3.2, where linear feature extraction methods like principal component analysis (PCA) and linear discriminant analysis (LDA) were mentioned. Linear methods are easy to understand and are very simple to implement, but the linearity assumption does not hold in many real world scenarios. Images of handwritten digits do not conform to the linearity assumption [113]; rotation, shearing, and variation of stroke widths can at best be approximated by linear functions only in a *small* neighborhood (as in the use of tangent distance [68]). A transformation as simple as translating an object on a uniform background cannot be represented as a linear function of the pixels. This has motivated the design of nonlinear mapping methods in a general setting. Note, however, that a globally nonlinear mapping can often be approximated by a linear mapping in a local region. In fact, this is the essence of many of the algorithms considered in this chapter.

In this chapter, we shall survey some of the recent nonlinear dimensionality reduction algorithms, with an emphasis on several algorithms that perform nonlinear mapping via the notion of learning the data manifold. Since we are mostly interested in unsupervised learning, supervised nonlinear dimensionality methods such as hierarchical discriminant regression (HDR) [118] are omitted from this survey. Some of the methods considered in this chapter have also been surveyed recently in [284] and [34].

2.1 Overview

The history of nonlinear mapping is long, tracing back to Sammon's mapping in 1969 [223]. Over time, different techniques have been proposed, such as projection pursuit [93] and projection pursuit regression [92], self organizing maps (SOM) [152], principal curve and its extensions [107, 249, 239, 144], auto-encoder neural networks [7, 57], generative topographic maps (GTM) [24], and kernel principal component analysis [228]. A comparison of some of these methods can be found in [180].

A new line of nonlinear mapping algorithms has been proposed recently based on the notion of manifold learning. Given a data set that is assumed to be lying approximately on a (Riemannian)

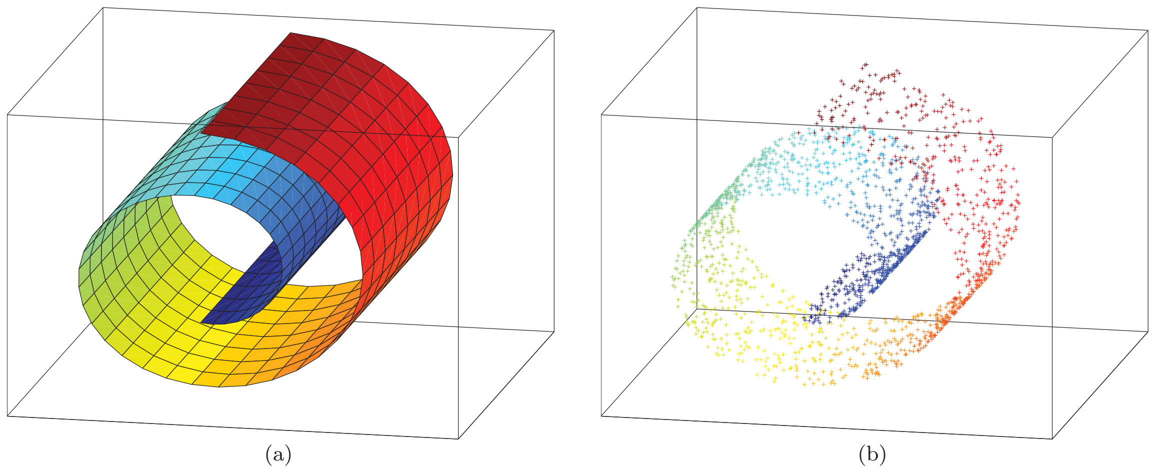


Figure 2.1: An example of a manifold. This example is usually known as the “Swiss roll”. (a) Surface of the manifold. (b) Data points lying on the manifold.

manifold in a high dimensional space, dimensionality reduction can be achieved by constructing a mapping that respects certain properties of the manifold. Isometric feature mapping (ISOMAP) [248], locally linear embedding (LLE), Laplacian eigenmap [16], semidefinite embedding [268], charting [29], and co-ordination-based ideas [220, 257] are some of the examples. The utility of manifold learning has been demonstrated in different applications, such as face pose detection [103, 172], face recognition [283, 276], analysis of facial expressions [75, 38], human motion data interpretation [133], gait analysis [75, 74], visualization of fiber traces [32], and wood texture analysis [196].

In this chapter, we shall review some of these algorithms, with an emphasis towards the manifold-based nonlinear mapping algorithms. It is hoped that this exposition can help the reader to become familiar with these recent exciting developments in nonlinear dimensionality reduction. Table 2.1 provides a comparison of the algorithms we are going to discuss. We want to point out that there are many other interesting manifold-related ideas that have been omitted in this chapter. Examples include stochastic embedding [112], locality preserving projections [109], Hessian eigenmap [67], semidefinite embedding [268] and its extension [267], the co-ordination type methods described in [257], [134] and [285], as well as the method in [31] which is related to Laplacian eigenmap. Robust statistics techniques can be used too [214]. It is also possible to learn a Parzen window along the data manifold [260].

The rest of this chapter is organized as follows. We first define our notation and describe some properties of a manifold in Section 2.2. Sammon’s mapping, probably the earliest nonlinear mapping algorithm, is discussed in Section 2.3. Auto-associative neural network [7], also known as auto-encoder neural networks [57], is described in Section 2.4. Kernel PCA is described in Section 2.5, followed by ISOMAP in Section 2.6, LLE in Section 2.7, and Laplacian eigenmap in Section 2.8. Three closely related ideas that involve combining different local co-ordinates are described in Section 2.9. We then show some results of running these algorithms on simple data sets in Section 2.10. Finally, we summarize our survey in Section 2.11.

2.2 Preliminary

Let $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ be the high-dimensional data set, where $\mathbf{y}_i \in \mathbb{R}^D$ and D is usually large. Let $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]$ be the $D \times n$ data matrix. We seek a transformation of $\mathcal{Y}$ that maps $\mathbf{y}_i$ to

	Key idea	Key Computation	Iterative	Parameters	Manifold
Sammon [223]	Minimize Sammon's stress	Gradient descent	Yes	none	No
auto-associative neural networks [7, 57]	Neural network for feature extraction and data reconstruction	Neural network training	Yes	none	No
KPCA [228]	PCA in feature space	eigenvectors of a large, full matrix	No	kernel function	No
ISOMAP [248]	Preserve geodesic distances	all pair shortest path; eigenvectors of a large, full matrix	No	neighborhood	Yes
LLE [226]	Same reconstruction weights	eigenvectors of a large, sparse matrix	No	neighborhood	Yes
Laplacian eigenmap [16]	Smooth graph embedding	eigenvectors of a large, sparse matrix	No	neighborhood; width parameter	Yes
Global coordination [220]	Mixture of factor analyzers; Unimodal posterior of global coordinate	EM algorithm derived by variational principle	Yes	Number of local models	Yes
Charting [29]	Nearby Gaussians are similar; Global coordinate by least square	constrained linear equations; eigenvectors of a small, full matrix	No	neighborhood (see caption)	Yes
LLC [247]	Local models are given; Global coordinate by LLE criterion	generalized eigenvectors of a small, full matrix	No	a mixture model fitted to the data	Yes

Table 2.1: A comparison of nonlinear mapping algorithms reviewed in this chapter. The dimensionality of the low dimensional space is assumed to be known, though some of these algorithms can also estimate it. "Neighborhood" refers to $N(\mathbf{x}_i)$ defined in section 2.2. If an algorithm is inspired from the notion of a manifold, we put an entry of "yes" in the "Manifold" column. Note that the neighborhood in charting [29] can be estimated from the data instead of having to be specified by the user. A matrix is "large" if its size is n by n, where n is the size of the data set. Only the few leading or trailing eigenvectors are needed if eigen-decomposition is performed.

its low dimensional counterpart $\mathbf{x}_i$, where $\mathbf{x}_i \in \mathbb{R}^d$ and d is small. Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ be the $d \times n$ matrix. We shall assume that different $\mathbf{y}_i$ do not lie randomly in $\mathbb{R}^D$, but approximately on a manifold, which is denoted by $\mathcal{M}$. The manifold may simply be a hyperplane, or it can be more complicated. An example of a "curved" manifold with the data points lying on it can be seen in Figure 2.1. This manifold assumption is reasonable because many real world phenomena are driven by a small number of latent factors. The high dimensional feature vectors observed are the results of applying a (usually unknown) mapping to the latent factors, followed by the introduction of noise. Consequently, high dimensional vectors in practice lie approximately on a low dimensional manifold.

Strictly speaking, what we refer to as "manifold" in this thesis should properly be called "Riemannian manifold." A Riemannian manifold is smooth and differentiable, and contains the notion of length. We leave the precise definition of Riemannian manifold to encyclopedias like Mathworld[1] and Wikipedia[2], and describe only some of its properties here. Every $\mathbf{y}$ in the manifold $\mathcal{M}$ has a neighborhood $N(\mathbf{y})$ that is homeomorphic[3] to a set S, where S is either an open subset of $\mathbb{R}^d$, or an open subset on the closed half of $\mathbb{R}^d$.

This mapping $\phi_{\mathbf{y}} : N(\mathbf{y}) \mapsto S$ is called a co-ordinate chart, and $\phi_{\mathbf{y}}(\mathbf{y})$ is called the "co-ordinate" of $\mathbf{y}$. A collection of co-ordinate charts that covers the entire $\mathcal{M}$ is called an atlas. If $\mathbf{y}$ is in two co-ordinate charts $\phi_{\mathbf{y}_1}$ and $\phi_{\mathbf{y}_2}$, $\mathbf{y}$ will have two (local) co-ordinates $\phi_{\mathbf{y}_1}(\mathbf{y})$ and $\phi_{\mathbf{y}_2}(\mathbf{y})$. These two co-ordinates should be "consistent" in the sense that there is a map to convert between $\phi_{\mathbf{y}_1}(\mathbf{y})$ and $\phi_{\mathbf{y}_2}(\mathbf{y})$, and the map is continuous for any path in $N(\mathbf{y}_1) \cap N(\mathbf{y}_2)$. For any $\mathbf{y}_i$ and $\mathbf{y}_j$ in $\mathcal{M}$, there can be many paths in $\mathcal{M}$ that connect $\mathbf{y}_i$ and $\mathbf{y}_j$. The shortest of such paths is called the geodesic[4] between $\mathbf{y}_i$ and $\mathbf{y}_j$. For example, the geodesic between two points on a sphere is an arc of a "great circle": a circle whose center coincides with the center of the sphere (Figure 2.2). The length of the geodesic between $\mathbf{y}_i$ and $\mathbf{y}_j$ is the geodesic distance between $\mathbf{y}_i$ and $\mathbf{y}_j$.

To perform nonlinear mapping, one can assume that there exists a mapping $\phi_{\text{global}}(.)$ that maps *all* points on $\mathcal{M}$ to $\mathbb{R}^d$. The "global co-ordinate" of $\mathbf{y}$, denoted by $\mathbf{x} = \phi_{\text{global}}(\mathbf{y})$, is regarded as the low dimensional representation of $\mathbf{y}$. In general, such a mapping may not exist[5]. In that case, a mapping that preserves a certain property of the manifold can be constructed to obtain $\mathbf{x}$.

Many of the nonlinear mapping algorithms that are manifold-based require a concrete definition of $N(\mathbf{y}_i)$, the neighborhood of $\mathbf{y}_i$. Two definitions are commonly used. In ϵ-neighborhood, $\mathbf{y}_j \in N(\mathbf{y}_i)$ if $||\mathbf{y}_i - \mathbf{y}_j|| < \epsilon$, where the norm is the Euclidean distance in $\mathbb{R}^D$. In $k$nn-neighborhood, $\mathbf{y}_j \in N(\mathbf{y}_i)$ if $\mathbf{y}_j$ is one of the k nearest neighbors of $\mathbf{y}_i$ in $\mathcal{Y}$, or vice versa. In both cases, ϵ or k is a user-defined parameter. $k$nn neighborhood has the advantage that it is independent of the scale of the data, though it can lead to too small a neighborhood when the number of data points is large. Note that the neighborhood can be defined in a data-driven manner [29] instead of being specified by a user.

2.3 Sammon's mapping

Sammon's mapping [223], which is an example of metric least square scaling [49], is perhaps the most well-known algorithm for nonlinear mapping. Sammon's mapping is an algorithm for multidimen-

[1] `http://mathworld.wolfram.com`

[2] `http://en2.wikipedia.org/`

[3] Two (topological) spaces are homeomorphic if there exists a continuous and invertible function between the two spaces, and that the inverse function is also continuous.

[4] Strictly speaking, geodesics are curves with zero covariant derivatives of their velocity vectors along the curve. A shortest curve must be a geodesic, whereas a geodesic might not be a shortest curve.

[5] For example, there is no such map (homeomorphism) between all points on a sphere and $\mathbb{R}^2$. However, if we exclude the north pole of a sphere, we can construct such a mapping.

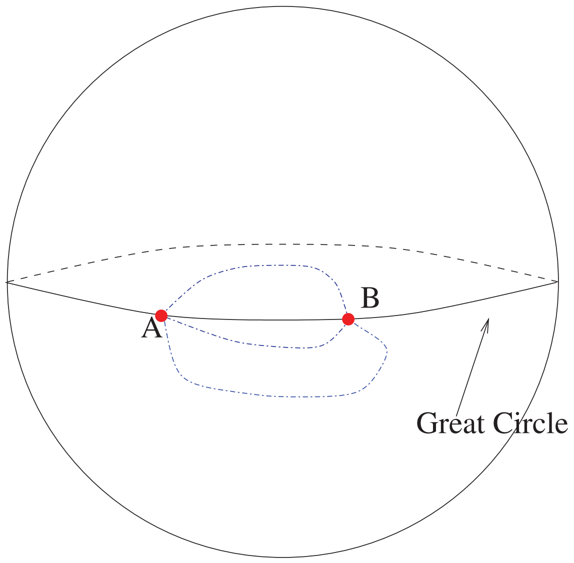


Figure 2.2: An example of a geodesic. For two points A and B on the sphere, many lines (the dash-dot lines) can be drawn to connect them. However, the shortest of these lines, which is the solid line joining A and B, is called the geodesic between A and B. In the case of a sphere, the geodesic is simply the great circle.

sional scaling and it maps a set of n items into an Euclidean space based on the dissimilarity values. This problem is related to the metric embedding problem considered by theoretical computer scientists [119]. Sammon's mapping can be used for dimensionality reduction if the dissimilarity matrix is based on the Euclidean distance between the data points in the high dimensional space.

Given a n by n matrix of dissimilarity values $\{\delta_{ij}\}$, where δ_{ij} denotes the dissimilarity between the i-th and the j-th items, we want to map the n items to n points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ in a low dimensional space, such that the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is as "close" to δ_{ij} as possible. Many different definitions of closeness have been proposed, with the "Sammon's stress", defined by Sammon, being the most popular. The Sammon's stress S is defined by

$$S = \sum_{i<j} \frac{(d_{ij} - \delta_{ij})^2}{\delta_{ij}} \Big/ \sum_{i<j} \delta_{ij}, \tag{2.1}$$

where $d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$ is the distance between $\mathbf{x}_i$ and $\mathbf{x}_j$. The quantity $(d_{ij} - \delta_{ij})^2$ measures the discrepancy between the observed dissimilarities with the actual distances. It is weighted by δ_{ij}^{-1} because if the dissimilarity is large, we should be more tolerant to the discrepancy. The division by $\sum_{i<j} \delta_{ij}$ makes S scale free. Sammon proposed the following iterative equation to find $\mathbf{x}_i$ that minimize S

$$x_{ik}^{\text{new}} = x_{ik} - \text{MF} \frac{\partial S}{\partial x_{ik}} \Big/ \Big| \frac{\partial^2 S}{\partial x_{ik}^2} \Big|, \tag{2.2}$$

where MF is a "magic factor", usually set to 0.3 or 0.4. Now, differentiating $d_{ij}^2 = \sum_{k'} (x_{ik'} - x_{jk'})^2$

with respect to x_{ik}, we get

$$\begin{aligned}2d_{ij}\partial d_{ij} &= 2(x_{ik}-x_{jk})\partial x_{ik}\\ \frac{\partial d_{ij}}{\partial x_{ik}} &= \frac{x_{ik}-x_{jk}}{d_{ij}}.\end{aligned}$$

So, the gradient of S is

$$\begin{aligned}\frac{\partial S}{\partial x_{ik}} &= \Big(\frac{2}{\sum_{i<j}\delta_{ij}}\Big)\sum_{j=1,j\neq i}^{m}\frac{d_{ij}-\delta_{ij}}{d_{ij}\delta_{ij}}(x_{ik}-x_{jk})\\ &= \Big(\frac{2}{\sum_{i<j}\delta_{ij}}\Big)\sum_{j=1,j\neq i}^{m}\left(\frac{1}{\delta_{ij}}-\frac{1}{d_{ij}}\right)(x_{ik}-x_{jk}),\end{aligned} \tag{2.3}$$

where x_{ik} is the k-th component in $\mathbf{x}_i$. For the second order information, note that

$$\begin{aligned}&\left(\frac{2}{\sum_{i<j}\delta_{ij}}\right)^{-1}\frac{\partial}{\partial x_{uv}}\frac{\partial S}{\partial x_{ik}}\\ =&I(v=k)\sum_{j=1,j\neq i}^{m}\left(\frac{1}{\delta_{ij}}-\frac{1}{d_{ij}}\right)(I(u=i)-I(u=j))\\ &+\sum_{j=1,j\neq i}^{m}\frac{1}{d_{ij}^2}(x_{ik}-x_{jk})\left(I(u=i)\frac{x_{iv}-x_{jv}}{d_{ij}}+I(u\neq i)I(u=j)\frac{x_{jv}-x_{iv}}{d_{ij}}\right)\\ =&I(v=k)\left(I(u=i)\sum_{j=1,j\neq i}^{m}\left(\frac{1}{\delta_{ij}}-\frac{1}{d_{ij}}\right)-I(u\neq i)\left(\frac{1}{\delta_{iu}}-\frac{1}{d_{iu}}\right)\right)\\ &+I(u=i)\sum_{j=1,j\neq i}^{m}\frac{(x_{ik}-x_{jk})(x_{iv}-x_{jv})}{d_{ij}^3}+I(u\neq i)\frac{(x_{ik}-x_{uk})(x_{uv}-x_{iv})}{d_{iu}^3}\end{aligned} \tag{2.4}$$

where $I(.)$ is the indicator function defined as

$$I(\text{true})=1 \qquad I(\text{false})=0.$$

One can use a nonlinear optimization algorithm other than Equation (2.2) to minimize S. It is also possible to implement Sammon's mapping by a feed-forward neural network [180] or in an incremental manner [129]. Note that Sammon's mapping is "global" and considers all the interpoint distances between the n items. This can be a drawback for data like the Swiss roll data set, where Euclidean distances between pairs of points that are far away from each other do not reveal the true structure of the data.

2.4 Auto-associative neural network

A special type of feed-forward neural network, "auto-associative neural network" [7, 57], can be used for nonlinear dimensionality reduction. An example of such a network is shown in Figure 2.3. The idea is to model the functional relationship between $\mathbf{x}_i$ and $\mathbf{y}_i$ by a neural network. If $\mathbf{x}_i$ is a good representation for $\mathbf{y}_i$, it should contain sufficient information to reconstruct $\mathbf{y}_i$ via a neural

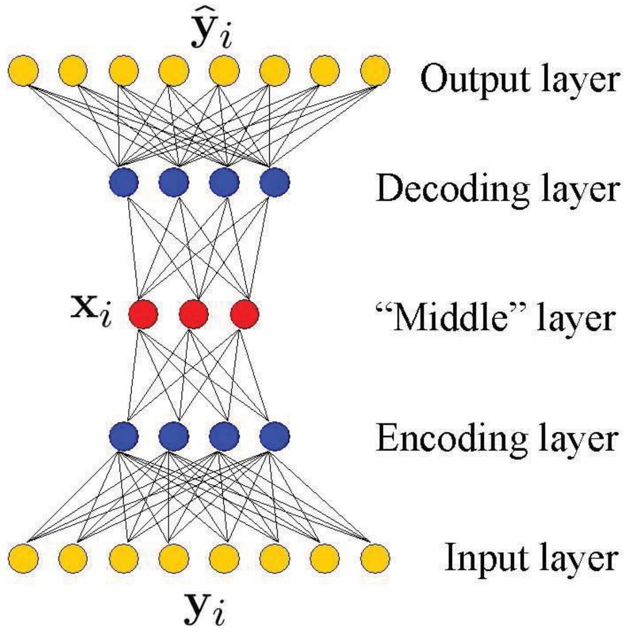


Figure 2.3: Example of an auto-associative neural network. This network extracts $\mathbf{x}_i$ with 3 features from the given data $\mathbf{y}_i$ with 8 features.

network (decoding network), with the "decoding layer" as its hidden layer. To obtain $\mathbf{x}_i$ from $\mathbf{y}_i$, another neural network (encoding network) is needed, with the "encoding layer" as its hidden layer. The encoding network and the decoding network are connected so that the output of the encoding network is used as the input of the decoding network, and both of them correspond to $\mathbf{x}_i$. The high-dimensional data points $\mathbf{y}_i$ are used as both the input and the target for training in this neural network. Sum of square error can be used as the objective function for training. Note that the neural network in Figure 2.3 is just an example; alternative architecture can be used. For example, multiple hidden layers can be used, and the number of neurons in the encoding and decoding layers can also be different.

The advantage of this approach is that mapping a new $\mathbf{y}$ to the corresponding $\mathbf{x}$ is easy: just feed $\mathbf{y}$ to the neural network and extract the output of the encoding layer. Also, there exists a number of software packages for training neural networks. The drawback is that it is difficult to determine the appropriate network architecture to best reduce the dimension for any given data set. Also, training of a neural network involves an optimization problem that is considerably more difficult than the eigen-decomposition required by some other nonlinear mapping methods like ISOMAP, LLE, or Laplacian eigenmap, which we shall examine later in this chapter.

2.5 Kernel PCA

The basic idea of kernel principal component analysis (KPCA) is to transform the input patterns to an even higher dimensional space nonlinearly and then perform principal component analysis in the new space. It is inspired from the success of the support vector machines (SVM) [189].

2.5.1 Recap of SVM

Consider a mapping $\phi : \mathbb{R}^D \mapsto \mathcal{H}$, where $\mathcal{H}$ is a Hilbert space. $\mathcal{H}$ can be, for example, a (very) high dimensional Euclidean space. By convention, $\mathbb{R}^D$ and $\mathcal{H}$ are called the input space and the feature space, respectively. The point $\mathbf{y}_i$ in $\mathbb{R}^D$ is first transformed into the Hilbert space $\mathcal{H}$ by $\phi(\mathbf{y}_i)$. SVM assumes a suitable transformation $\phi(.)$ such that the transformed data set is more linearly separable in $\mathcal{H}$ than in $\mathbb{R}^D$, and a large margin classifier in $\mathcal{H}$ is trained to separate the transformed data. It turns out that the large margin classifier can be trained by using only the inner product between the transformed data $\langle \phi(\mathbf{y}_i), \phi(\mathbf{y}_j) \rangle$, without knowing $\phi(.)$ explicitly. Therefore, in practice, the kernel function $K(\mathbf{y}_i, \mathbf{y}_i)$ is specified instead of $\phi(.)$, where

$$K(\mathbf{y}_i, \mathbf{y}_i) = \langle \phi(\mathbf{y}_i), \phi(\mathbf{y}_j) \rangle.$$

Specifying the kernel function $K(.,.)$ instead of the mapping $\phi(.)$ has the advantage of computational efficiency when $\mathcal{H}$ is of high dimension. Also, this allows us to generalize to infinite dimensional $\mathcal{H}$, which happens when the radial basis function kernel is used. This use of kernel function to replace an explicit mapping is often called "the kernel trick". Intuitively, the kernel function, being an inner product, represents the similarity between $\mathbf{y}_i$ and $\mathbf{y}_j$.

The kernel trick can be illustrated by the following example with $D = 2$. Let $\phi(\mathbf{y}_i) \equiv (y_{i1}^2, \sqrt{2} y_{i1} y_{i2}, y_{i2}^2)^T$, where $\mathbf{y}_i = (y_{i1}, y_{i2})^T$. The kernel function corresponding to this $\phi(.)$ is $K(\mathbf{y}_i, \mathbf{y}_j) = (y_{i1} y_{j1} + y_{i2} y_{j2})^2$, because

$$\begin{aligned} K(\mathbf{y}_i, \mathbf{y}_j) &= (y_{i1} y_{j1} + y_{i2} y_{j2})^2 \\ &= y_{i1}^2 y_{j1}^2 + 2 y_{i1} y_{j1} y_{i2} y_{j2} + y_{i2}^2 y_{j2}^2 \\ &= (y_{i1}^2, \sqrt{2} y_{i1} y_{i2}, y_{i2}^2)(y_{j1}^2, \sqrt{2} y_{j1} y_{j2}, y_{j2}^2)^T \\ &= \phi(\mathbf{y}_i)^T \phi(\mathbf{y}_j). \end{aligned}$$

Many different kernel functions have been proposed. Polynomial kernel, defined as $K(\mathbf{y}_i, \mathbf{y}_j) = (\mathbf{y}_i^T \mathbf{y}_j + 1)^r$ with r as the parameter (degree) of the kernel, corresponds to a polynomial decision boundary in the input space. The radial basis function (RBF) kernel is defined by $K(\mathbf{y}_i, \mathbf{y}_j) = \exp(\omega ||\mathbf{y}_i - \mathbf{y}_j||^2)$, where ω is the width parameter. SVM classifiers using RBF kernel are related to RBF neural networks, except that for SVM, the centers of the basis functions and the corresponding weights are estimated by the quadratic programming solver simultaneously [229]. The choice of the appropriate kernel function in an application is difficult in general. This is still an active research area, with many principles being proposed [121, 154, 227].

2.5.2 Kernel PCA

One important lesson we can learn from SVM is that a linear algorithm in the feature space corresponds to a nonlinear algorithm in the input space. Different types of nonlinearity can be achieved by different kernel functions. Kernel PCA [228] utilizes this to generalize PCA to become nonlinear. For ease of notation, we shall assume $\mathcal{H}$ is of finite dimension[6].

KPCA follows the steps of the standard PCA, except the data set under consideration is

[6]The case for infinite dimensional $\mathcal{H}$ is similar, with operators replacing matrices and eigenfunctions replacing eigenvectors.

$\{\phi(\mathbf{y}_1), \ldots, \phi(\mathbf{y}_n)\}$. Let $\tilde{\phi}(\mathbf{y}_i)$ be the "centered" version of $\phi(\mathbf{y}_i)$,

$$\tilde{\phi}(\mathbf{y}_i) = \phi(\mathbf{y}_i) - \frac{1}{n}\sum_{i=1}^{n} \phi(\mathbf{y}_i).$$

The covariance matrix $\mathbf{C}$ is given by

$$\mathbf{C} = \frac{1}{n}\sum_{i} \tilde{\phi}(\mathbf{y}_i)\tilde{\phi}(\mathbf{y}_i)^T.$$

The eigenvalue problem $\lambda\mathbf{v} = \mathbf{C}\mathbf{v}$ is solved to find the (kernel) principal component $\mathbf{v}$. Because

$$\mathbf{v} = \frac{1}{\lambda}\mathbf{C}\mathbf{v} = \frac{1}{\lambda n}\sum_{i} \tilde{\phi}(\mathbf{y}_i)\left(\tilde{\phi}(\mathbf{y}_i)^T\mathbf{v}\right),$$

$\mathbf{v}$ is in the subspace spanned by $\tilde{\phi}(\mathbf{y}_i)$, and it can be written as

$$\mathbf{v} = \sum_{j} \alpha_j \tilde{\phi}(\mathbf{y}_j).$$

Denote $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$. Let $\tilde{\mathbf{K}}$ be the symmetric matrix such that its (i,j)-th entry $\tilde{K}_{ij}$ is $\tilde{\phi}(\mathbf{y}_i)^T\tilde{\phi}(\mathbf{y}_j)$. Rewrite $\lambda\mathbf{v} = \mathbf{C}\mathbf{v}$ as

$$\lambda\sum_{j} \alpha_j\tilde{\phi}(\mathbf{y}_j) = \frac{1}{n}\sum_{ij}\alpha_j\tilde{K}_{ij}\tilde{\phi}(\mathbf{y}_i). \tag{2.5}$$

By multiplying both sides with $\tilde{\phi}(\mathbf{y}_l)^T$, we have

$$\lambda\sum_{j}\alpha_j\tilde{K}_{lj} = \frac{1}{n}\sum_{ij}\alpha_j\tilde{K}_{ij}\tilde{K}_{li} \qquad \forall l, \tag{2.6}$$

which, in matrix form, can be written as

$$\lambda n\tilde{\mathbf{K}}\boldsymbol{\alpha} = \tilde{\mathbf{K}}^2\boldsymbol{\alpha}. \tag{2.7}$$

Since $\tilde{\mathbf{K}}$ is symmetric, $\tilde{\mathbf{K}}$ and $\tilde{\mathbf{K}}^2$ have the same set of eigenvectors. This set of eigenvectors is also the solution to the generalized eigenvalue problem in Equation (2.7). Therefore, $\boldsymbol{\alpha}$, and hence $\mathbf{v}$, can be found by solving $\lambda\boldsymbol{\alpha} = \tilde{\mathbf{K}}\boldsymbol{\alpha}$. For projection purposes, it is customary to normalize $\mathbf{v}$ to norm one. Since $||\mathbf{v}||^2 = \boldsymbol{\alpha}^T\tilde{\mathbf{K}}\boldsymbol{\alpha}$, we should divide $\boldsymbol{\alpha}$ by $\sqrt{\boldsymbol{\alpha}^T\tilde{\mathbf{K}}\boldsymbol{\alpha}}$. To perform dimensionality reduction for $\mathbf{y}$, it is first mapped to the feature space as $\tilde{\phi}(\mathbf{y})$, and its projection on $\mathbf{v}$ is given by

$$\tilde{\phi}(\mathbf{y})^T\mathbf{v} = \sum_{i}\tilde{\phi}(\mathbf{y})^T\alpha_i\tilde{\phi}(\mathbf{y}_i) = \boldsymbol{\alpha}^T\tilde{\mathbf{k}}_{\mathbf{y}}, \tag{2.8}$$

where $\tilde{\mathbf{k}}_{\mathbf{y}} = \left(\tilde{\phi}(\mathbf{y})^T\tilde{\phi}(\mathbf{y}_1), \ldots, \tilde{\phi}(\mathbf{y})^T\tilde{\phi}(\mathbf{y}_n)\right)^T$. Finally, by rewriting the relationship

$$\begin{aligned}
\tilde{K}_{ij} &= \tilde{\phi}(\mathbf{y}_i)^T\tilde{\phi}(\mathbf{y}_j) \\
&= \left(\phi(\mathbf{y}_i) - \frac{1}{n}\sum_{l=1}^{n}\phi(\mathbf{y}_l)\right)^T \left(\phi(\mathbf{y}_j) - \frac{1}{n}\sum_{k=1}^{n}\phi(\mathbf{y}_k)\right) \\
&= \phi(\mathbf{y}_i)^T\phi(\mathbf{y}_j) - \frac{1}{n}\sum_{l=1}^{n}\phi(\mathbf{y}_l)^T\phi(\mathbf{y}_j) \\
&\qquad - \frac{1}{n}\sum_{k=1}^{n}\phi(\mathbf{y}_k)^T\phi(\mathbf{y}_i) + \frac{1}{n^2}\sum_{k=1}^{n}\sum_{l=1}^{n}\phi(\mathbf{y}_k)^T\phi(\mathbf{y}_k)
\end{aligned}$$

in matrix form, we have

$$\tilde{\mathbf{K}} = \mathbf{H}_n\mathbf{K}\mathbf{H}_n, \tag{2.9}$$

where $\mathbf{H}_n = \mathbf{I} - \frac{1}{n}\mathbf{1}_{n,n}$ is a centering matrix with $\mathbf{1}_{n,n}$ denoting a matrix of size n by n with all entries one, and $\mathbf{K}$ is the kernel matrix with its (i, j)-th entry given by $K(\mathbf{y}_i, \mathbf{y}_j)$. A similar expression can be derived for $\tilde{\phi}(\mathbf{y})^T\tilde{\phi}(\mathbf{y}_j)$.

KPCA solves the eigenvalue problem of a n by n matrix, which may be larger than the D by D matrix considered by PCA. Recall D is the dimension of $\mathbf{y}_i$. The number of possible features to be extracted in KPCA can be larger than D. This contrasts with the standard PCA, where at most D features can be extracted. An interesting problem related to KPCA is how to map $\mathbf{z}$, the projection of $\phi(\mathbf{y})$ into the subspace spanned by the first few kernel principal components, back to the input space. This can be useful for, say, image denoising with KPCA [185]. The search for the "best" $\mathbf{y'}$ such that $\phi(\mathbf{y'}) \approx \mathbf{z}$ is known as the pre-image problem and different solutions have been proposed [160, 5].

In summary, KPCA consists of the following steps.

1. Let $\mathbf{K}$ be the kernel matrix, where $K_{ij} = \phi(\mathbf{y}_i, \mathbf{y}_j)$. Compute $\tilde{\mathbf{K}}$ by

$$\tilde{\mathbf{K}} = \mathbf{H}_n\mathbf{K}\mathbf{H}_n.$$

2. Solve the eigenvalue problem $\lambda\boldsymbol{\alpha} = \tilde{\mathbf{K}}\boldsymbol{\alpha}$ and find the eigenvectors corresponding to the largest few eigenvalues.

3. Normalize $\boldsymbol{\alpha}$ by dividing it by $\sqrt{\boldsymbol{\alpha}^T\tilde{\mathbf{K}}\boldsymbol{\alpha}}$.

4. For any $\mathbf{y}$, its projection to a principal component can be found by $\boldsymbol{\alpha}^T\tilde{\mathbf{k}}_{\mathbf{y}}$, where

$$\tilde{\mathbf{k}}_{\mathbf{y}} = \mathbf{H}_n(\mathbf{k}_{\mathbf{y}} - \frac{1}{n}\mathbf{K}\mathbf{1}_{n,1}),$$

$\mathbf{k}_{\mathbf{y}} = (K(\mathbf{y}, \mathbf{y}_1), \ldots, K(\mathbf{y}, \mathbf{y}_n)$ and $\mathbf{1}_{n,1}$ is a n by 1 vector with all entries equal to one.

2.6 ISOMAP

The basic idea of isometric feature map (ISOMAP) [248] is to find a mapping that best preserves the geodesic distances between any two points on a manifold. Recall that the geodesic distance between two points on a manifold is defined as the length of the shortest path on the manifold that connects the two points. ISOMAP constructs a mapping from $\mathbf{y}_i$ to the $\mathbf{x}_i$ ($\mathbf{x}_i \in \mathbb{R}^d$) such that the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathbb{R}^d$ is as close as possible to the geodesic distance between $\mathbf{y}_i$ and $\mathbf{y}_j$ on the manifold.

Geodesic distances are hard enough to find when the manifold is known, let alone in the current case where the manifold is unknown and only points on the manifold are given. So, ISOMAP approximates the geodesic distances by first constructing a neighborhood graph to represent the manifold. The vertex v_i in the neighborhood graph $G = (V, E)$ corresponds to the high dimensional data point $\mathbf{y}_i$. An edge $e(i, j)$ between v_i and v_j exists if and only if $\mathbf{y}_i$ is in the neighborhood of $\mathbf{y}_j$, $N(\mathbf{y}_j)$, and the weight of this edge is $||\mathbf{y}_i - \mathbf{y}_j||$. Details of $N(\mathbf{y}_j)$ are described in section 2.2. An example of a neighborhood graph is shown in Figure 2.4(b) for the data shown in Figure 2.4(a). ISOMAP approximates a path on the manifold by a path in the neighborhood graph. The geodesic between $\mathbf{y}_i$ and $\mathbf{y}_j$ corresponds to the shortest path between v_i and v_j. The estimation problem of the geodesic distances between all pairs of points $\mathbf{y}_i$ and $\mathbf{y}_j$ thus becomes the all-pairs shortest path problem in the neighborhood graph. It can be solved [46] either by the Floyd-Warshall algorithm, or by Dijkstra's algorithm with different source vertices. The latter is more efficient because the neighborhood graph is sparse. An example of how the shortest path approximates the geodesic is shown in Figure 2.4(c). It can be shown that the shortest path distances converge to the geodesic distances asymptotically [18].

The next step of ISOMAP finds $\mathbf{x}_i$ that best preserve the geodesic distances. Let g_{ij} denote the estimated geodesic distance between $\mathbf{y}_i$ and $\mathbf{y}_j$, and write $\mathbf{G} = \{\tilde{g}_{ij}\}$ as the geodesic distance matrix. The optimal $\mathbf{x}_i$ can be found by applying the classical scaling [49], a simple multi-dimensional scaling technique. Let $d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||$. Without loss of generality, assume $\sum_i \mathbf{x}_i = \mathbf{0}$. We have the following:

$$\begin{aligned}
d_{ij}^2 &= (\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_j) = ||\mathbf{x}_i||^2 + ||\mathbf{x}_j||^2 - 2\mathbf{x}_i^T\mathbf{x}_j \\
\sum_i d_{ij}^2 &= \sum_i ||\mathbf{x}_i||^2 + n||\mathbf{x}_j||^2 \\
\sum_{ij} d_{ij}^2 &= 2n \sum_i ||\mathbf{x}_i||^2 \\
\text{So, } \sum_i ||\mathbf{x}_i||^2 &= \frac{1}{2n} \sum_{ij} d_{ij}^2 \\
||\mathbf{x}_j||^2 &= \frac{1}{n} \sum_i d_{ij}^2 - \frac{1}{2n^2} \sum_{ij} d_{ij}^2
\end{aligned}$$

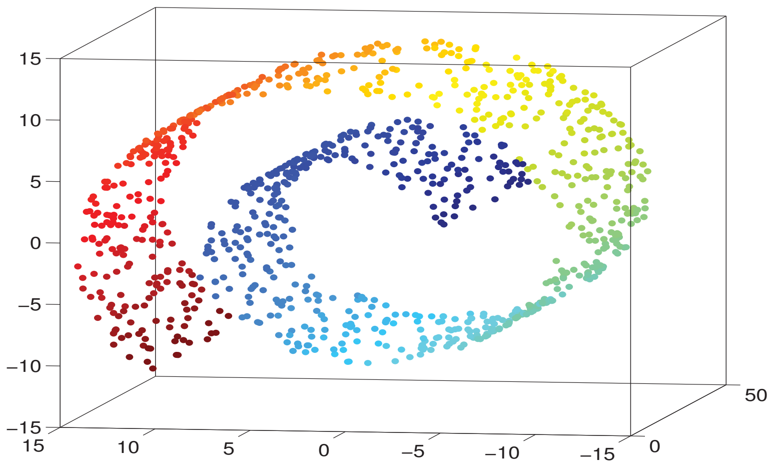
and

$$2\mathbf{x}_i^T\mathbf{x}_j = \frac{1}{n} \sum_j d_{ij}^2 + \frac{1}{n} \sum_i d_{ij}^2 - \frac{1}{n^2} \sum_{ij} d_{ij}^2 - d_{ij}^2. \tag{2.10}$$

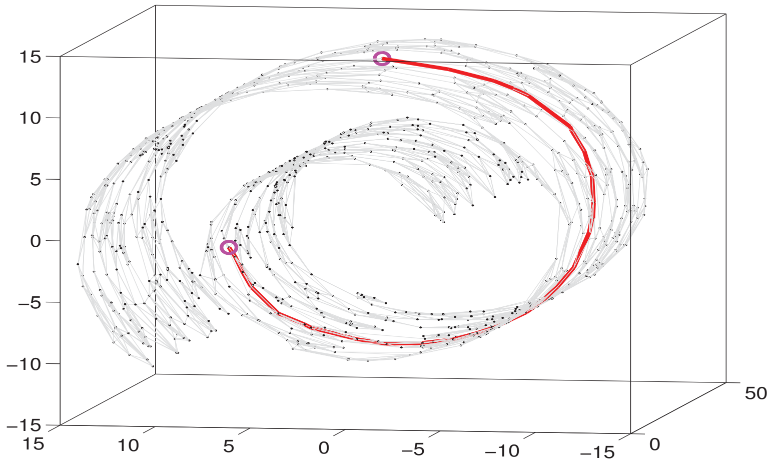
If we replace d_{ij} with the estimated geodesic distance g_{ij} in Equation (2.10), b_{ij}, the target inner product between $\mathbf{x}_i$ and $\mathbf{x}_j$, is given by

$$b_{ij} = \frac{1}{2}\left(\frac{1}{n} \sum_j g_{ij}^2 + \frac{1}{n} \sum_i g_{ij}^2 - \frac{1}{n^2} \sum_{ij} g_{ij}^2 - g_{ij}^2\right). \tag{2.11}$$

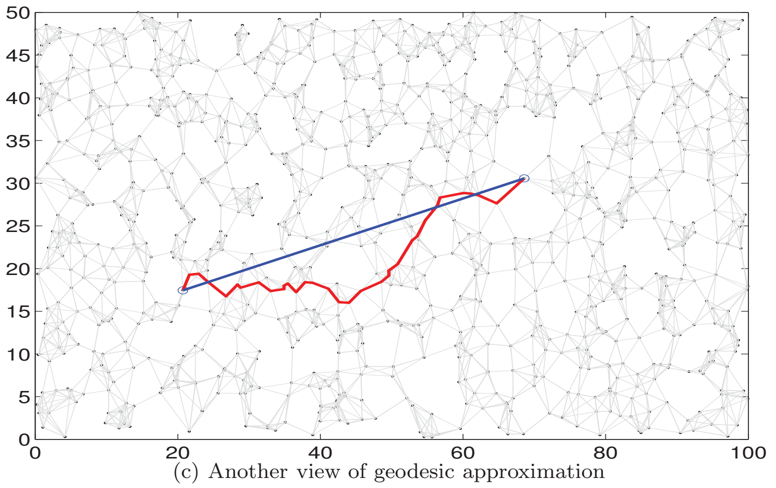
Let $\mathbf{A} = \{a_{ij}\}$ with $a_{ij} = -\frac{1}{2} g_{ij}^2$. Equation (2.11) means that $\mathbf{B} = \mathbf{H}_n \mathbf{A} \mathbf{H}_n$, where $\mathbf{B} = \{b_{ij}\}$, $\mathbf{H}_n = \mathbf{I} - \frac{1}{n}\mathbf{1}_{n,n}$ and $\mathbf{1}_{n,n}$ denotes a n by n matrix with all entries one.



(a) Input data



(b) Neighborhood graph and geodesic approximation



(c) Another view of geodesic approximation

Figure 2.4: Example of neighborhood graph and geodesic distance approximation. (a) Input data. (b) The neighborhood graph and an example of the shortest path. (c) This is the same as (b), except the manifold is flattened. The true geodesic (blue line) is approximated by the shortest path (red line).

Computing $\mathbf{H}_n\mathbf{A}\mathbf{H}_n$ is effectively a centering operation on $\mathbf{A}$, i.e., each column is subtracted by its corresponding column mean, and each row is subtracted by its corresponding row mean. Because multiplication of $\mathbf{H}_n$ has this effect of "zeroing" the means for different rows and columns, $\mathbf{H}_n$ is often referred to as the centering matrix. The centering operation is also seen in other embedding algorithm such as KPCA (section 2.5). Since $\mathbf{B}$ is the matrix of target inner product, we have $\mathbf{B} \approx \mathbf{X}^T\mathbf{X}$, where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$. We recover $\mathbf{X}$ by finding the best rank-d approximation for $\mathbf{B}$, which can be obtained via the eigen-decomposition of $\mathbf{B}$. Let $\lambda_1, \ldots, \lambda_d$ be the d largest eigenvalues of $\mathbf{B}$ with corresponding eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$. We have $\mathbf{X} = [\sqrt{\lambda_1}\mathbf{v}_1, \ldots, \sqrt{\lambda_d}\mathbf{v}_d]^T$. Here, we assume $\lambda_i > 0$ for all $i = 1, \ldots, d$. Unlike Sammon's mapping, the objective function for the optimal $\mathbf{X}$ is less explicit: it is the sum of the square error (squared Frobenius norm) between the target inner product (b_{ij}) and the actual inner product $(\mathbf{x}_i^T\mathbf{x}_j)$.

One drawback of ISOMAP is the $O(n^2)$ memory requirement for storing the dense matrix of geodesic distances. Also, solving the eigenvalue problem of a large dense matrix is relatively slow. To reduce both the computational and memory requirements, landmark ISOMAP [55] sets apart a subset of $\mathcal{Y}$ as landmark points and preserves only the geodesic distances from $\mathbf{y}_i$ to these landmark points. A similar idea has been applied to Sammon's mapping before [25]. A continuum version of ISOMAP has also been proposed [282]. ISOMAP can fail when there is a "hole" in the manifold [66]. We also want to note that an exact isometric mapping of a manifold is theoretically possible only when the manifold is "flat", i.e., when the curvature tensor is zero, as pointed out in [16].

To summarize, ISOMAP consists of the following steps:

1. Construct a neighborhood graph using either the ϵ neighborhood or the $k$nn neighborhood.

2. Solve the all pair shortest path problem on the neighborhood graph to obtain an estimate of the geodesic distances g_{ij}.

3. Compute $\mathbf{A} = \{a_{ij}\}$, where $a_{ij} = -\frac{1}{2}g_{ij}^2$, and $\mathbf{B} = \mathbf{H}_n\mathbf{A}\mathbf{H}_n$.

4. The d largest eigenvalues and the corresponding eigenvectors of $\mathbf{B}$ are found and $\mathbf{X} = [\sqrt{\lambda_1}\mathbf{v}_1, \ldots, \sqrt{\lambda_d}\mathbf{v}_d]^T$.

2.7 Locally Linear Embedding

In locally linear embedding (LLE) [219, 226], each local region on a manifold is approximated by a linear hyperplane. LLE maps the high dimensional data points into a low dimensional space so that the local geometric properties, represented by the reconstruction weights, are best preserved.

Specifically, $\mathbf{y}_i$ is reconstructed by its projection $\hat{\mathbf{y}}_i$ on the hyperplane H passing through its neighbors $N(\mathbf{y}_i)$ (defined in section 2.2). Mathematically,

$$\mathbf{y}_i \approx \hat{\mathbf{y}}_i = \sum_j w_{ij}\mathbf{y}_j,$$

with the constraint $\sum_j w_{ij} = 1$ to reflect the translational invariance for the reconstruction. By minimizing the sum of square error of this approximation, we can also achieve invariance for rotation and scaling. The weights w_{ij} reflect the local geometric properties of $\mathbf{y}_i$. This interpretation on w_{ij}, however, is reasonable only when $\mathbf{y}_i$ is well approximated by $\hat{\mathbf{y}}_i$, i.e., when $\mathbf{y}_i$ is close to H.

The weights are found by solving the following optimization problem:

$$\min_{\{w_{ij}\}} ||\mathbf{y}_i - \sum_j w_{ij}\mathbf{y}_j||^2 \quad \text{subject to } \sum_j w_{ij} = 1, w_{ij} = 0 \text{ if } \mathbf{y}_j \notin N(\mathbf{y}_i) \qquad \text{for all } i. \tag{2.12}$$

Now, write $N(\mathbf{y}_i) = \{\mathbf{y}_{\tau_1}, \ldots, \mathbf{y}_{\tau_L}\}$ and denote $\mathbf{z}_j = \mathbf{y}_{\tau_j}$. Note that $\mathbf{y}_i \notin N(\mathbf{y}_i)$. The optimization problem (2.12) can be solved efficiently by first constructing a L by L matrix $\mathbf{F}$ such that $f_{jk} = (\mathbf{z}_j - \mathbf{x}_i)^T(\mathbf{z}_k - \mathbf{x}_i)$. Equivalently, $\mathbf{F} = (\mathbf{Z} - \mathbf{x}_i\mathbf{1}_{1,L})^T(\mathbf{Z} - \mathbf{x}_i\mathbf{1}_{1,L})$, where $\mathbf{F} = \{f_{jk}\}$, $\mathbf{1}_{1,L}$ is a 1 by L vector with all entries one, and $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_l]$. The next step is to solve the equation

$$\mathbf{F}\mathbf{u} = \mathbf{1}_{l,1} \tag{2.13}$$

and then we normalize[7] the solution $\mathbf{u}$ by $\tilde{u}_j = u_j / \sum_j u_j$. The values of $\tilde{u}_j$ are assigned to the corresponding w_{ij}, i.e., $w_{i,\tau_j} = u_j$, and the rest of w_{ij} are set to zero. Sometimes, $\mathbf{F}$ can be singular. This can happen when the neighborhood size L is larger than D, the dimension of $\mathbf{y}_i$. In this case, a small regularization term $\delta\mathbf{I}_L$ is added to $\mathbf{F}$ before solving the Equation (2.13). This regularization has the effect of preferring values of w_{ij} with small $\sum_j w_{ij}^2$. Finding u_j is efficient because only small linear systems of equations are solved. Note that u_j can be negative and $\hat{\mathbf{y}}_i$ can be outside the convex hull of $N(\mathbf{y}_i)$.

In the second phase of LLE, we seek $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ such that $\mathbf{x}_i \approx \sum_j w_{ij}\mathbf{x}_j$, and $\mathbf{x}_i \in \mathbb{R}^d$. To make the problem well-defined, additional constraints $\sum_i \mathbf{x}_i = \mathbf{0}$ and $\sum_i \mathbf{x}_i\mathbf{x}_i^T = \mathbf{I}_d$ are needed. The second constraint has the effect of both fixing the scale and enforcing different features in $\mathbf{x}_i$ to carry independent information by requiring the sample covariances between different variables in $\mathbf{x}_i$ to be zero. The optimization problem is now

$$\min_{\{\mathbf{x}_i\}} ||\mathbf{x}_i - \sum_j w_{ij}\mathbf{x}_j||^2 \qquad \text{subject to } \sum_i \mathbf{x}_i = \mathbf{0} \text{ and } \sum_i \mathbf{x}_i\mathbf{x}_i^T = \mathbf{I}_d. \tag{2.14}$$

Note the similarity between Equations (2.12) and (2.14). Let $\mathbf{x}^{(i)}$ denote the i-th row of $\mathbf{X}$. Equation (2.14) can be rewritten as

$$\min_{\mathbf{X}} \quad \text{trace}(\mathbf{X}(\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})\mathbf{X}^T) \qquad \text{subject to } \mathbf{1}_{1,m}\mathbf{x}^{(i)} = 0 \text{ and } \mathbf{x}^{(i)T}\mathbf{x}^{(i)} = \delta_{ij}. \tag{2.15}$$

This can be solved by eigen-decomposition on $\mathbf{M} = (\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})$. Note that $\mathbf{M}$ is positive semi-definite. Let $\mathbf{v}_j$ be the eigenvector corresponding to the $(j+1)$-th *smallest* eigenvalue. The optimal $\mathbf{X}$ is given by $\mathbf{X} = [\mathbf{v}_1, \ldots, \mathbf{v}_d]^T$. The first constraint is automatically satisfied because $\mathbf{1}_{n,1}$ is the eigenvector of $\mathbf{M}$ with eigenvalue 0. This eigenvalue problem is relatively easy because $\mathbf{M}$ is sparse and can be represented as a product of sparser matrices $(\mathbf{I} - \mathbf{W})^T$ and $(\mathbf{I} - \mathbf{W})$.

The above exposition of LLE assumes the pattern matrix as input. LLE can be modified to work with a dissimilarity matrix [226]. There is also a supervised extension of LLE [53, 54], which uses the class labels to modify the neighborhood structure. The kernel trick can also be applied to LLE to visualize the data points in the feature space [56]. The case when LLE is applied to data sets with natural clustering structure has been examined in [206].

In summary, LLE includes the following steps:

[7]The normalization is valid because $\sum_j u_j = \mathbf{1}_{1,m}\mathbf{F}^{-1}\mathbf{1}_{m,1}$ and hence $\sum_j u_j$ cannot be zero, by the positive definiteness of $\mathbf{F}^{-1}$.

1. Find the neighbors of each $\mathbf{y}_i$ according to either ϵ-neighborhood or $k$nn neighborhood.
2. For each $\mathbf{y}_i$, form the matrix $\mathbf{F}$ and solve the equation $\mathbf{F}\mathbf{u} = \mathbf{1}_{L,1}$. After normalizing $\mathbf{u}$ by $\tilde{u}_j = u_j / \sum_j u_j$, set $w_{i,\tau_j} = \tilde{u}_j$ and the remaining w_{ij} to zero.
3. Find the second to the $(d+1)$-th smallest eigenvalues of $(\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})$ by a sparse eigenvalue solver and let $\{\mathbf{v}_1, \ldots, \mathbf{v}_d\}$ be the eigenvectors.
4. Obtain the reduced dimension representation by $\mathbf{X} = [\mathbf{v}_1, \ldots, \mathbf{v}_d]^T$.

2.8 Laplacian Eigenmap

The approach taken by Laplacian eigenmap [16] for nonlinear mapping is different from those of ISOMAP and LLE. Laplacian eigenmap constructs orthogonal smooth functions defined on the manifold based on the Laplacian of the neighborhood graph. It has its roots in spectral graph theory [42].

As in ISOMAP, a neighborhood graph $G = (V, E)$ is first constructed. Unlike ISOMAP, where the weight w_{ij} of the edge (v_i, v_j) represents the distance between v_i and v_j, the weight in Laplacian eigenmap represents the similarity between v_i and v_j. The weight w_{ij} can be set by

$$w_{ij} = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{4t}\right), \tag{2.16}$$

with t as an algorithmic parameter, or it can be simply set to one. The use of the exponential function to transform a distance value to a similarity value can be justified by its relationship to the heat kernel [16].

The nonlinear mapping problem is recast as the graph embedding problem that maps the vertices in the neighborhood graph G to $\mathbb{R}^d$. The first step is to find a "good" function $f(.) : V \mapsto \mathbb{R}$ that maps the vertices in G to a real number. Since the domain of $f(.)$ is finite, $f(.)$ can be represented by a vector $\mathbf{u}$, with $f(v_i) = u_i$. According to spectral graph theory, the smoothness of f can be defined by

$$S \equiv \frac{1}{2} \sum_{ij} w_{ij}(u_i - u_j)^2. \tag{2.17}$$

The intuition of S is that, for large w_{ij}, the vertices v_i and v_j are "similar" and hence the difference between $f(v_i)$ and $f(v_j)$ should be small if $f(.)$ is smooth. A smooth mapping $f(.)$ is desirable because a faithful embedding of the graph should assign similar values to v_i and v_j when they are close. We can rewrite S as

$$\begin{aligned} S &= \frac{1}{2} \sum_{ij} (w_{ij} u_i^2 + w_{ij} u_j^2 - 2 u_i u_j) \\ &= \frac{1}{2} (\sum_i u_i^2 \sum_j w_{ij} + \sum_j u_j^2 \sum_i w_{ij} - 2 \sum_{ij} w_{ij} u_i u_j) \\ &= \sum_i u_i^2 \sum_j w_{ij} - \sum_{ij} w_{ij} u_i u_j = \mathbf{u}^T \mathbf{L} \mathbf{u}, \end{aligned} \tag{2.18}$$

where $\mathbf{L}$ is the graph Laplacian defined by $\mathbf{L} = \mathbf{D} - \mathbf{W}$, $\mathbf{W} = \{w_{ij}\}$ is the graph weight matrix, and $\mathbf{D}$ is a diagonal matrix with $d_{ii} = \sum_j w_{ij}$. The matrix $\mathbf{L}$ can be thought of as the Laplacian operator on functions defined on the graph. Since d_{ii} can be interpreted as the importance of v_i, the natural inner product between two functions $f_1(.)$ and $f_2(.)$ defined on the graph is $\langle f_1, f_2 \rangle = \mathbf{u}_1^T \mathbf{D} \mathbf{u}_2$. Because the constant function is the smoothest and is uninteresting, we seek $f(.)$ to be as smooth as possible while being orthogonal to the constant function. The norm of $f(.)$ is constrained to be one to make the problem well-defined. Thus we want to solve

$$\min_{\mathbf{u}} \mathbf{u}^T \mathbf{L} \mathbf{u} \qquad \text{subject to } \mathbf{u}^T \mathbf{D} \mathbf{u} = 1 \text{ and } \mathbf{u}^T \mathbf{D} \mathbf{1}_{n,1} = 0. \tag{2.19}$$

This can be done by solving the generalized eigenvalue problem

$$\mathbf{L}\mathbf{u} = \lambda \mathbf{D} \mathbf{u}, \tag{2.20}$$

after noting that $\mathbf{1}_{n,1}$ is a solution to Equation (2.20) with $\lambda = 0$. Here, $\mathbf{1}_{n,1}$ denotes a n by 1 vector with all entries one. As $\mathbf{L}$ is positive semi-definite, the eigenvector corresponding to the second smallest eigenvalue of Equation (2.20) yields the desired $f(.)$. In general, d orthogonal[8] functions $\{f_1(.), \ldots, f_d(.)\}$ that are as smooth as possible are sought to map the vertices to $\mathbb{R}^d$. The functions can be obtained by the eigenvectors corresponding to the second to the $(d+1)$-th smallest eigenvalues in Equation (2.20). The low dimensional representation of $\mathbf{y}_i$ is then given by $\mathbf{x}_i = (f_1(v_i), f_2(v_i), \ldots, f_d(v_i))^T$. In matrix form, $\mathbf{X} = [\mathbf{u}_1, \ldots, \mathbf{u}_d]^T$.

The embedding problem of the neighborhood graph and the embedding problem of the points in the manifold is related in the following way. A smooth function $f(.)$ that maps the point $\mathbf{y}_i$ in the manifold to $\mathbf{x}_i \in \mathbb{R}^d$ is preferable, because a faithful mapping should give similar values (small $||\mathbf{x}_i - \mathbf{x}_j||$) to $\mathbf{y}_i$ and $\mathbf{y}_j$ when $||\mathbf{y}_i - \mathbf{y}_j||$ is small. A small $||\mathbf{y}_i - \mathbf{y}_j||$ corresponds to a large w_{ij} in the graph. Thus, intuitively, a smooth function defined on the graph corresponds to a smooth function defined on the manifold. In fact, this relationship can be made more rigorous, because the graph Laplacian is closely related to the Laplace-Beltrami operator on the manifold, which in turn is related to the smoothness of a function defined on the manifold. The eigenvectors of the graph Laplacian correspond to the eigenfunctions of the Laplace-Beltrami operator, and the eigenfunctions with small eigenvalues provide a “smooth” basis of the functions defined on the manifold. The neighborhood graph used in Laplacian eigenmap can thus be viewed as a discretization tool for computation on the manifold.

There is also a close relationship between Laplacian eigenmap and spectral clustering. In fact, the spectral clustering algorithm in [194] is almost the same as first performing Laplacian eigenmap and then applying k-means clustering on the low dimensional feature vectors. The manifold structure discovered by Laplacian eigenmap can also be used to train a classifier in a semi-supervised setting [182]. The Laplacian of a graph can also lead to an interesting kernel function (as in SVM) for vertices in a graph [154]. This idea of nonlinear mapping via graph embedding has also been extended to learn a linear mapping [109] as well as generalized to the case when a vector is associated with each vertex in the graph [31].

To sum up, the steps for Laplacian eigenmap include:

1. Construct a neighborhood graph of $\mathcal{Y}$ by either the ϵ-neighborhood or the $k$nn neighborhood.

2. Compute the edge weight w_{ij} by either $\exp(||\mathbf{y}_i - \mathbf{y}_j||^2/(4t))$, or simply set w_{ij} to 1.

3. Compute $\mathbf{D}$ and the graph Laplacian $\mathbf{L}$.

[8]Orthogonality is preferred as it suggests the independence of information. Also, in PCA, each of the extracted features is orthogonal to the others.

4. Find the second to the $(d+1)$-th smallest eigenvalues in the generalized eigenvalue problem $\mathbf{Lu} = \lambda \mathbf{Du}$ and denote the eigenvectors by $\mathbf{u}_1, \ldots, \mathbf{u}_d$. The low dimensional feature vectors are given by $\mathbf{X} = [\mathbf{u}_1, \ldots, \mathbf{u}_d]^T$.

2.9 Global Co-ordinates via Local Co-ordinates

Recall that in section 2.2, an atlas of a manifold $\mathcal{M}$ is defined as a collection of co-ordinate charts that covers the entire $\mathcal{M}$, and overlapping charts can be "connected" smoothly. This idea has inspired several nonlinear mapping algorithms [220, 29, 247] which construct different local charts and join them together.

There are two stages in these type of algorithms. First, different local models are fitted to the data, usually by the means of a mixture model. Each local model gives rise to a local co-ordinate system. A local model can be, for example, a Gaussian or a factor analyzer. Let $\mathbf{z}_{is}$ be the local co-ordinate given to $\mathbf{y}_i$ by the s-th local co-ordinate system. Let r_{is} denote the suitability of using the s-th local model for $\mathbf{y}_i$. We require $r_{is} \geq 0$ and $\sum_s r_{is} = 1$. The introduction of r_{is} can represent the fact that only a small number of local models are meaningful for each $\mathbf{y}_i$. Typically, r_{is} is obtained as the posterior probability of the s-th local model, given $\mathbf{y}_i$.

In the second stage, different local co-ordinates of $\mathbf{y}_i$ are combined to give a global co-ordinate. Let $\mathbf{g}_{is}$ be the global co-ordinate of $\mathbf{y}_i$ due to the s-th local model, and let $\mathbf{g}_i \in \mathbb{R}^d$ be the corresponding "combined" global co-ordinate. In the three papers we have considered here, $\mathbf{g}_{is}$ is simply the affine transform of the local co-ordinate, $\mathbf{g}_{is} = \mathbf{L}_s \tilde{\mathbf{z}}_{is}$. Here, $\tilde{\mathbf{z}}_{is}$ is the "augmented" $\mathbf{z}_{ik}$, $\tilde{\mathbf{z}}_{is} = [\mathbf{z}_{ik}^T, 1]^T$. $\mathbf{L}_s$ is the (unknown) transformation matrix with d rows for the s-th local model. Note that it is desirable for neighboring local models to be "similar" so that the global co-ordinates are more consistent. An important characteristic of the algorithms in this section is that, unlike ISOMAP, LLE, or Laplacian eigenmap, extension for a point $\mathbf{y}$ that is outside the training data $\mathcal{Y}$ is easy after computing $\mathbf{z}_s$ and r_s for different s.

2.9.1 Global Co-ordination

In the global co-ordination algorithm in [220], the first and the second stages are performed simultaneously by the variational method. The first stage is done by fitting a mixture of factor analyzers. Under the s-th local model, a data point is modeled by

$$\mathbf{y}_i = \boldsymbol{\mu}_s + \boldsymbol{\Lambda}_s \mathbf{z}_{is} + \boldsymbol{\epsilon}_{is}, \tag{2.21}$$

where $\boldsymbol{\mu}_s$ is the mean, $\boldsymbol{\Lambda}_s$ is the factor loading matrix, and $\boldsymbol{\epsilon}_{is}$ is the noise that follows $\mathcal{N}(\mathbf{0}, \Psi_s)$, a multivariate Gaussian with mean $\mathbf{0}$ and covariance Ψ_s. By the definition of factor analyzer, Ψ_s is diagonal. The hidden variable $\mathbf{z}_{is}$ is assumed to follow $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The scale of $\mathbf{z}_{is}$ is unimportant because it can be absorbed by the factor loading matrix. Let α_s be the prior probability of the s-th factor analyzer. The parameters are $\{\alpha_s, \boldsymbol{\mu}_s, \boldsymbol{\Lambda}_s, \Psi_s\}$, and the data density is given by

$$\begin{aligned} p(\mathbf{y}_i) &= \sum_s \int_{\mathbf{z}_{is}} p(\mathbf{y}_i | s, \mathbf{z}_{is}) p(\mathbf{z}_{is} | s) P(s) d\mathbf{z}_{is} \\ &= \sum_s \alpha_s (2\pi)^{-D/2} \big(\det(\boldsymbol{\Lambda}_s \boldsymbol{\Lambda}_s^T + \Psi_s)\big)^{-1/2} \\ &\qquad \exp\Big(-\frac{1}{2}(\mathbf{y}_i - \boldsymbol{\mu}_s)^T (\boldsymbol{\Lambda}_s \boldsymbol{\Lambda}_s^T + \Psi_s)^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_s)\Big). \end{aligned} \tag{2.22}$$

We define r_{is} as the posterior probability of the s-th local model given $\mathbf{y}_i$, $P(s|\mathbf{y}_i)$, and it can be computed based on Equation (2.22). Equation (2.22) also gives rise to $p(\mathbf{z}_{is}|s, \mathbf{y}_i)$ and hence $p(\mathbf{g}_{is}|s, \mathbf{y}_i)$, because $\mathbf{g}_{is}$ is a function of $\mathbf{z}_{is}$ and $\mathbf{L}_s$. The posterior probability of the global co-ordinate is defined as

$$p(\mathbf{g}_i|\mathbf{y}_i) = \sum_s P(s|\mathbf{y}_i) p(\mathbf{g}_{is}|s, \mathbf{y}_i). \tag{2.23}$$

Equation (2.23) assumes that the overall global co-ordinate $\mathbf{g}_i$ is selected among different $\mathbf{g}_{is}$, with s stochastically selected according to the posterior probability of the s-th model. In the case where $\mathbf{y}_i$ is likely to be generated either by the j-th or the k-th local model, the corresponding global co-ordinates $\mathbf{g}_{ij}$ and $\mathbf{g}_{ik}$ should be similar. This implies that the posterior density $p(\mathbf{g}_i|\mathbf{y}_i)$ should be unimodal. Enforcing the unimodality of $p(\mathbf{g}_i|\mathbf{y}_i)$ directly is difficult. So, the authors in [220] instead drive $p(\mathbf{g}_i|\mathbf{y}_i)$ to be as similar to a Gaussian distribution as possible by adding an extra term to the log-likelihood objective function to be maximized:

$$\Phi = \sum_i \log p(\mathbf{y}_i) - \sum_{is} D_{KL}\big(q(\mathbf{g}_i, s|\mathbf{y}_i) || p(\mathbf{g}_i, s|\mathbf{y}_i)\big). \tag{2.24}$$

Here, $D_{KL}(Q||P)$ is the Kullback-Leibler divergence defined as

$$D_{KL}(Q||P) = \int Q(\mathbf{y}) \log \frac{Q(\mathbf{y})}{P(\mathbf{y})} \, d\mathbf{y}, \tag{2.25}$$

and $q(\mathbf{g}_i, s|\mathbf{y}_i)$ is assumed to be factorized as

$$q(\mathbf{g}_i, s|\mathbf{y}_i) = q_i(\mathbf{g}_i|\mathbf{x}_i) q_i(s|\mathbf{y}_i)$$

with $q_i(\mathbf{g}_i|\mathbf{y}_i)$ as a Gaussian and $q_i(s|\mathbf{y}_i)$ as a multinomial distribution. This addition of a divergence term between a posterior distribution and a factorized distribution is commonly seen in the literature on the variational method. The objective function in Equation (2.24) can be maximized by an EM-type algorithm, which estimates the parameters $\{\alpha_s, \boldsymbol{\mu}_s, \boldsymbol{\Lambda}_s, \Psi_s, \mathbf{L}_s\}$ as well as the parameters for $q_i(\mathbf{g}_i|\mathbf{y}_i)$ and $q_i(s|\mathbf{y}_i)$. Since the first and the second stages are carried out simultaneously, local models that lead to consistent global co-ordinates are implicitly favored.

2.9.2 Charting

For the charting algorithm in [29], the first and the second stages are performed separately. This decoupling decreases the complexity of the optimization problem and can reduce the chance of getting trapped in poor local minima. In the first stage, a mixture of Gaussians is fitted to the data,

$$p(\mathbf{y}) = \sum_s \alpha_s \mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s), \tag{2.26}$$

with the constraint that two adjacent Gaussians should be "similar". This is achieved by using a prior distribution on the mean vectors and the covariance matrices that encourages the similarity of adjacent Gaussians:

$$p(\{\boldsymbol{\mu}_s\}, \{\boldsymbol{\Sigma}_s\}) \propto \exp\big(-\sum_s \sum_{j, j\neq s} \lambda_s(\boldsymbol{\mu}_j) D_{KL}(\mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s) || \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j))\big), \tag{2.27}$$

where $\lambda_s(\boldsymbol{\mu}_j)$ measures the closeness between the locations of the s-th and the j-th Gaussian components. It is set to $\lambda_s(\boldsymbol{\mu}_j) \propto \exp(-||\boldsymbol{\mu}_s - \boldsymbol{\mu}_j||^2/(2\sigma^2))$, where σ is a width parameter determined according to the neighborhood structure. The prior distribution also makes the parameter estimation problem more well-conditioned. In practice, n Gaussian components are used, with the center of the i-th component $\boldsymbol{\mu}_i$ set to $\mathbf{y}_i$ and the weight of each component set to $1/n$. The only parameters to be estimated are the covariance matrices. The MAP estimate of the covariance matrices can be shown to satisfy a set of constrained linear equations and they are obtained by solving this set of equations.

In the second stage, the local co-ordinate $\mathbf{z}_{is}$ is first obtained as $\mathbf{z}_{is} = \mathbf{V}^T(\mathbf{x}_i - \boldsymbol{\mu}_s)$, where $\mathbf{V}$ consists of the d leading eigenvectors of $\boldsymbol{\Sigma}_s$. We can regard $\mathbf{z}_{is}$ as the feature extracted from $\mathbf{y}_i$ using PCA on the s-th local model. The local model weight r_{is} is, once again, set to the posterior probability of the s-th local model given $\mathbf{y}_i$. The transformation matrices $\mathbf{L}$ are found by solving the following weighted least square problem:

$$\min_{\{\mathbf{L}_s\}} \sum_{i,j,k} r_{ij} r_{ik} ||\mathbf{L}_j \tilde{\mathbf{z}}_{ij} - \mathbf{L}_k \tilde{\mathbf{z}}_{ik}||_F^2. \tag{2.28}$$

Here, $||\mathbf{X}||_F^2$ denotes the square of the Frobenius norm, $||\mathbf{X}||_F^2 \equiv \text{trace}(\mathbf{X}^T\mathbf{X})$. Intuitively, we want to find the transformation matrices such that the global co-ordinates due to different local models are the most consistent in the least square sense, weighted by the importance of different local models.

Equation (2.28) can be solved as follow. Let K and h be the number of local models and the length of the augmented local co-ordinate $\tilde{\mathbf{z}}_{is}$, respectively. Define $\tilde{\mathbf{Z}}_s = [\tilde{\mathbf{z}}_{1s}, \ldots, \tilde{\mathbf{z}}_{ns}]$ as the h by n matrix of local co-ordinates using the s-th local model for all the data points. Define the Kh by n matrix $\mathbf{T}_s$ by $\mathbf{T}_s = [\mathbf{0}_{n,(s-1)h}, \tilde{\mathbf{Z}}_s^T, \mathbf{0}_{n,(K-s)h}]^T$, where $\mathbf{0}_{n,m}$ denotes a zero matrix with size n by m. Let $\mathbf{P}_s$ be a n by n diagonal matrix where the (i,i)-th entry is r_{is}. The solution to Equation (2.28) is given by the d trailing eigenvectors of the Kh by Kh matrix $\mathbf{Q}\mathbf{Q}^T$, where $\mathbf{Q} = \sum_j \sum_{k,j\neq k} \big((\mathbf{T}_j - \mathbf{T}_k)\mathbf{P}_j\mathbf{P}_k\big)$. Note that the second stage is independent of the first stage. In particular, alternative collection of local models can be used, as long as $\mathbf{z}_{is}$ and r_{is} can be calculated.

2.9.3 LLC

The LLC algorithm described in [247] concerns the second stage only. Given the local co-ordinates $\mathbf{z}_{is}$ and the model confidences r_{is} computed from the first stage, the LLC algorithm finds the best $\mathbf{L}_s$ such that the local geometric properties are best preserved in the sense of the LLE loss function. The global co-ordinate $\mathbf{g}_i$ is assumed to be a weighted sum in the form

$$\mathbf{g}_i = \sum_s r_{is}\mathbf{g}_{is} = \sum_s r_{is}\mathbf{L}_s\tilde{\mathbf{z}}_{is}. \tag{2.29}$$

Suppose there are K local models, each of which gives a local co-ordinate $\mathbf{z}_{is}$ in a $h-1$ dimensional space[9]. We stack $\tilde{\mathbf{z}}_{is} r_{is}$ for different s to get a vector of length Kh, $\mathbf{u}_i = [r_{i1}\tilde{\mathbf{z}}_{i1}^T, r_{i2}\tilde{\mathbf{z}}_{i2}^T, \ldots, r_{iK}\tilde{\mathbf{z}}_{iK}^T]^T$, and concatenate different $\mathbf{L}_s$ to form a d by Kh matrix $\mathbf{J} = [\mathbf{L}_1, \mathbf{L}_2, \ldots, \mathbf{L}_K]$. (Each $\mathbf{L}_s$ is of size d by h.) Equation (2.29) can be rewritten as $\mathbf{g}_i = \mathbf{J}\mathbf{u}_i$. The global co-ordinate matrix, $\mathbf{G} = (\mathbf{g}_1, \ldots, \mathbf{g}_n)$, is thus given by $\mathbf{G} = \mathbf{J}\mathbf{U}$, where $\mathbf{U}$ is a Kh by n matrix $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n]$. Denote the i-th row of $\mathbf{J}$ by $\mathbf{j}^{(i)}$. If we substitute $\mathbf{G}$ as $\mathbf{Y}$ in the LLE

[9]In general, different local models can give local co-ordinates with different lengths, as emphasized in [247]. Here we assume a common h for the ease of notation.

objective function in equation (2.15), we have

$$\begin{aligned}&\min_{\mathbf{J}} \operatorname{trace}(\mathbf{JU}(\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})\mathbf{U}^T\mathbf{J}^T)\\ &\text{subject to } \mathbf{j}^{(i)}\mathbf{U}\mathbf{1}_{n,1}=0 \text{ and } \mathbf{j}^{(i)}\mathbf{U}\mathbf{U}^T\mathbf{j}^{(j)T}=\delta_{ij},\end{aligned} \tag{2.30}$$

where $\mathbf{W}$ is defined in the same way (the neighborhood reconstruction weight) as in section 2.7. Here, $\mathbf{1}_{n,1}$ denotes a n by 1 vector with all entries one. Note that obtaining $\mathbf{W}$ is efficient (see section 2.7 for details). The value of $\mathbf{j}^{(i)}$ can be obtained as the solution of the generalized eigenvalue problem $\big(\mathbf{U}(\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})\mathbf{U}^T\big)\mathbf{v} = \lambda(\mathbf{U}\mathbf{U}^T)\mathbf{v}$. The authors in [247] claim that the $\mathbf{j}^{(i)}$ thus obtained satisfies the constraint $\mathbf{j}^{(i)}\mathbf{U}\mathbf{1}_{m,1}=0$ automatically because $\mathbf{U}\mathbf{1}_{m,1}$ is an eigenvector of the generalized eigenvalue problem with eigenvalue 0. However, this is not true in general. In any case, the authors in [247] use the eigenvectors corresponding to the second to the $(d+1)$-th smallest eigenvalues as the solution of $\mathbf{J}$. Note that this generalized eigenvalue problem is about a Kh by Kh matrix, instead of a large n by n matrix in the original LLE. After finding $\mathbf{j}^{(i)}$, $\mathbf{J}$ and hence $\mathbf{L}_s$ are reconstructed. The global co-ordinate is obtained via equation (2.29).

The idea of this algorithm is somewhat analogous to the locality preserving projection (LPP) algorithm [109]. LPP simplifies the eigenvalue problem by the extra information that the projection should be linear, whereas the current algorithm simplifies the eigenvalue problem by the given mixture model.

2.10 Experiments

We applied some of these algorithms on three synthetic 3D data sets. The data manifold and the data points can be seen in Figure 2.5. The first data set, *parabolic*, consists of 2000 randomly sampled data points lying on a paraboloid. It is an example of a nonlinear manifold with a simple analytic form – a second degree polynomial in the co-ordinates in this case. The second data set *swiss roll* and the third data set *S-curve* are commonly used for validating manifold learning algorithms. Again, 2000 points are randomly sampled from the "Swiss roll" and the S-shaped surface to create the data sets, respectively. KPCA, ISOMAP, LLE, and Laplacian eigenmap were run on these 3D data sets to project the data to 2D. We have implemented KPCA and Laplacian eigenmap ourselves, while the implementations for ISOMAP[10] and LLE[11] were downloaded from their respective web sites. For ISOMAP, LLE, and Laplacian eigenmap, $k$nn neighborhood with $k = 12$ is used. The edge weight is set to one for Laplacian eigenmap. For KPCA, polynomial kernel with degree 2 is used. For comparison, the standard PCA and Sammon's mapping were also performed on these data sets. Sammon's mapping is initialized by the result of PCA.

The results of these algorithms can be seen in Figures 2.6, 2.7, and 2.8. The data points are colored differently to visualize their locations on the manifold. We intentionally omit the "goodness-of-fits" or "error" on the projection results, because the criteria used by different algorithms (Sammon's stress in Sammon's mapping, correlation of distances in ISOMAP, reconstruction error in LLE, residue variance in PCA and KPCA, to name a few) are very different and it can be misleading to compare them.

For the *parabolic* data set, we can see in Figures 2.6(b) and 2.6(c) that both ISOMAP and LLE recover the intrinsic co-ordinates very well, because the changes in the color of the data points after

[10]ISOMAP web site: `http://stanford.isomap.edu`
[11]LLE web site: `http://www.cs.toronto.edu/~roweis/lle/`

embedding are smooth. Since this manifold is quadratic, we expect that KPCA with a quadratic kernel function should also recover the true structure of the data. It turns out that the first two kernel principal components cannot lead to a clean mapping of the data points. Instead, the second and the third kernel principal components extract the structure of the data (Figure 2.6(a)). The first two features extracted by Laplacian eigenmap cannot recover the desired trend in the data. The target structure with slight distortion can be recovered if the second and the third extracted features are used instead (Figure 2.6(d)). PCA and Sammon's mapping cannot recover the structure of this data set (Figures 2.6(e) and 2.6(f)). The similarity of the results of PCA and Sammon's mapping can be attributed to the fact that Sammon's mapping is initialized by the PCA solution. The initial PCA solution is already a good solution with respect to Sammon's stress for this low-dimensional data set.

For the data set *swiss roll*, we can see from Figures 2.7(b) and 2.7(c) that ISOMAP and LLE performed a good job "unfolding" the manifold. For Laplacian eigenmap, once again, the first two extracted features cannot be interpreted easily, though the structure of the data set is revealed if the second and the third features are used (Figure 2.7(d)). KPCA cannot recover the intrinsic structure of the data set no matter which kernel principal component is used. An example of the poor result of KPCA is shown in Figure 2.7(a). PCA and Sammon's mapping also cannot recover the underlying structure (Figures 2.7(e) and 2.7(f)). The results for the third data set *S-curve* (Figure 2.8) are similar to those of *swiss roll*, with the exception that Laplacian eigenmap can recover the desired structure using the first two extracted features.

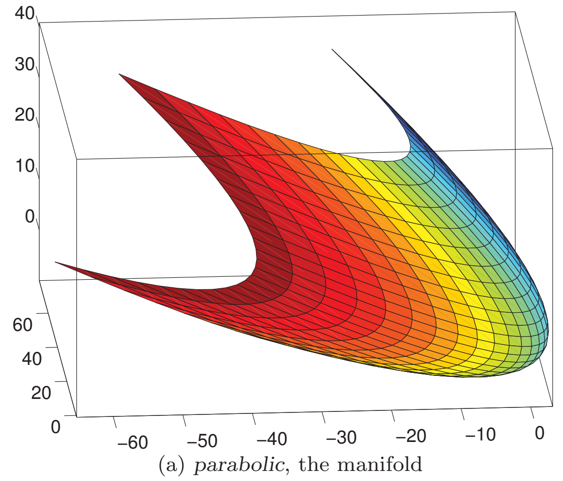
In addition to these synthetic data sets, we have also tested these nonlinear mapping algorithms on a high-dimensional real world data set: the face images used in [175] The task here is to classify a 64 by 64 face image in this data set as either the "Asian class" or the "non-Asian class". This data set will be described in more details in Section 3.3. The results of mapping these 4096D data points to 3D can be seen in Figure 2.9. Data points from the two classes are shown in different colors. The (training) error rates using quadratic discriminant analysis are also computed for different mappings. As we can see from Figures 2.9(a), 2.9(d), 2.9(e) and 2.9(f), the mapping results by Laplacian eigenmap, KPCA, PCA and Sammon's mapping are not very useful. The two classes are not well-separated, and the error rates are also high. ISOMAP maps the two classes more separately and has smaller error rates (Figure 2.9(b)). For LLE (Figure 2.9(c)), although the mapping results look a bit unnatural, the error rate turns out to be the smallest, indicating the two classes are reasonably separated. It should be noted that the intrinsic dimensionality of this data set is probably higher than 3. So, mapping the data to 3D, while good for visualization, can lose some information and is suboptimal for classification.

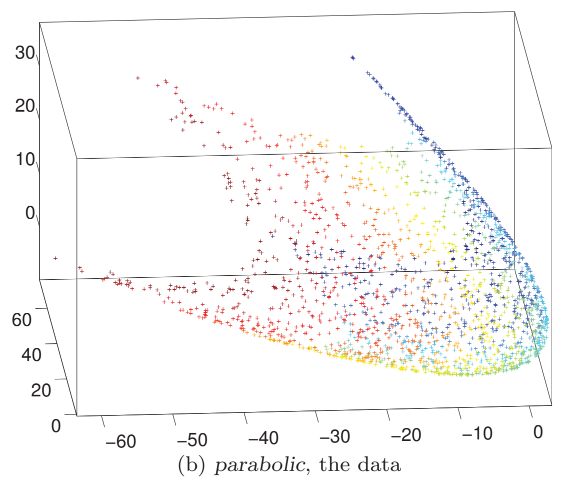
From these experiments, we can see that both ISOMAP and LLE recover the intrinsic structure of the data sets well. The performance of Laplacian eigenmap is less satisfactory. We have attempted to set the edge weight by the exponential function of distances (Equation (2.16)) instead of one, but the preliminary results suggest that a good choice of the width parameter t is hard to obtain. The standard PCA and Sammon's mapping cannot recover the target structure of the data. It is not surprising, because PCA is a linear algorithm and the underlying structure of the data cannot be reflected by any linear function of the features. For Sammon's mapping, it does not give very good results because Sammon's mapping is "global", meaning that the relationship between all pairs of data points in the 3D space is considered. Local properties of the manifold cannot be modeled. The reason for the failure of KPCA is that the parametric representation of the manifold for *swiss roll* and *S-curve* and the face images is hard to obtain, and is certainly not quadratic. So, the assumption in KPCA is violated and this leads to poor results.

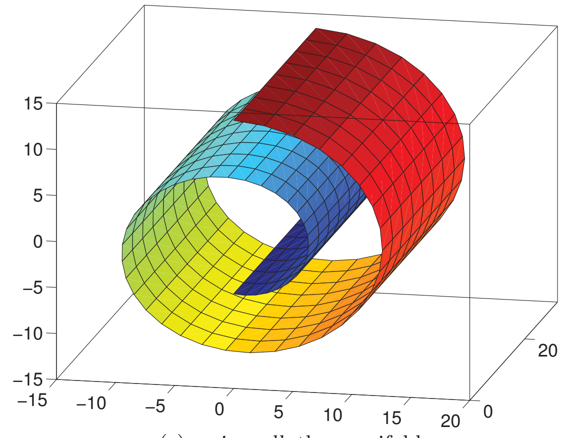
2.11 Summary

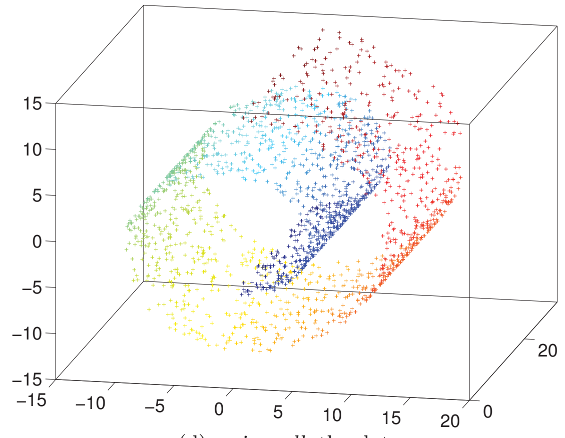
In this chapter, we have described different approaches for nonlinear mapping based on fairly different principles. The algorithms ISOMAP, LLE, and Laplacian eigenmap are non-iterative and require mainly eigen-decomposition, which is well understood with many off-the-shelf algorithms available. ISOMAP, LLE, and Laplacian eigenmap are basically non-parametric algorithms. While this provides extra flexibility to model the manifold, more data points are needed to give a good estimate of the low dimensional vector. The basic version of some of the algorithms (Sammon's mapping, ISOMAP, LLE, and Laplacian eigenmap) cannot generalize the mapping to patterns outside the training set $\mathcal{Y}$, though an out-of-sample extension has been proposed [17].

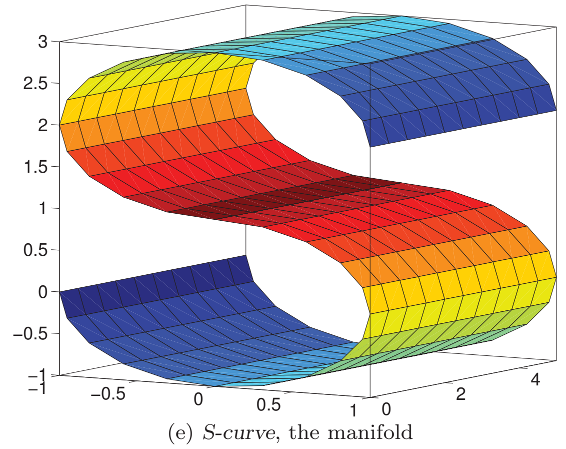
There are interesting connections between some of these algorithms. ISOMAP, LLE, and Laplacian Eigenmap can be shown to be the special cases of KPCA [105]. The matrix $\mathbf{M}$ in LLE can be shown to be related to the square of the Laplacian Beltrami operator [16], an important concept in Laplacian eigenmap. While these techniques have been successfully applied to high dimensional data sets like face images, digit images, texture images, motion data, and textual data, the relative merits of these algorithms in practice are still not clear. More comparative studies like the one in [196] would be helpful.


(a) *parabolic*, the manifold


(b) *parabolic*, the data


(c) *swiss roll*, the manifold


(d) *swiss roll*, the data


(e) *S-curve*, the manifold

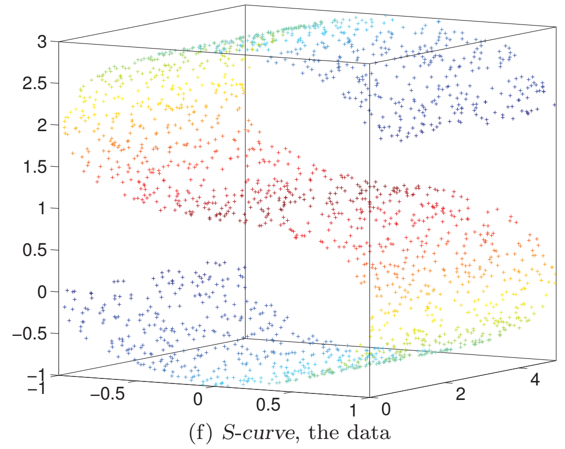
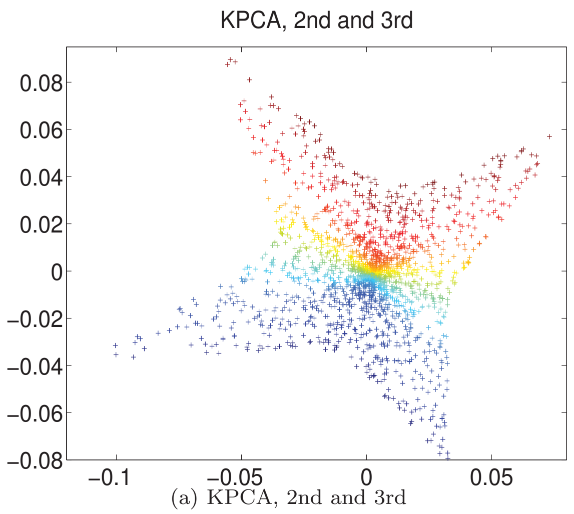
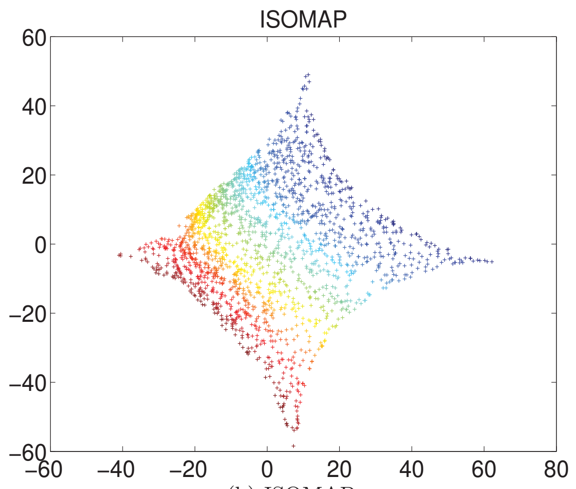

(f) *S-curve*, the data

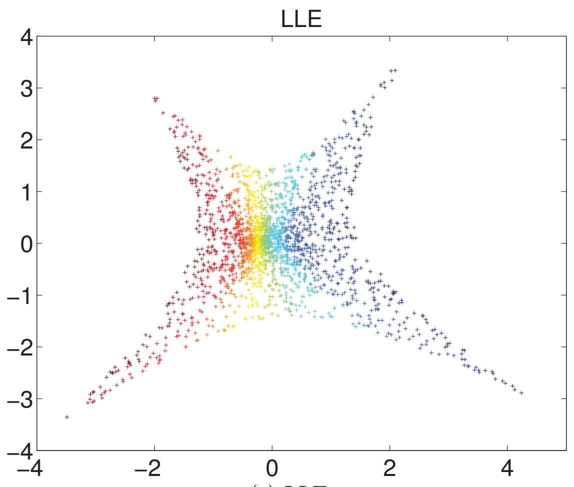
Figure 2.5: Data sets used in the experiments for nonlinear mapping. The manifold and the data points are shown. The data points are colored according to the major structure of the data as perceived by human.



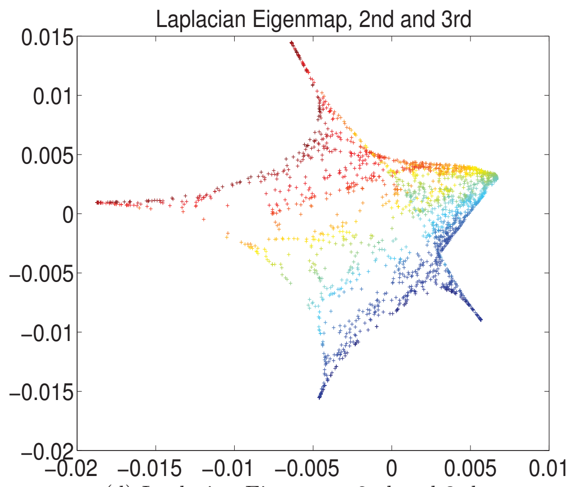
(a) KPCA, 2nd and 3rd



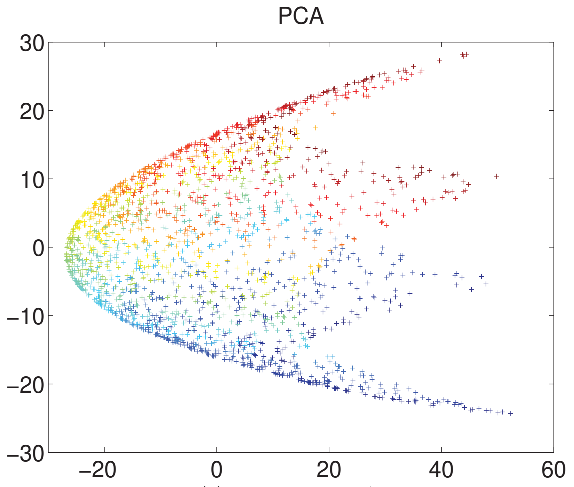
(b) ISOMAP



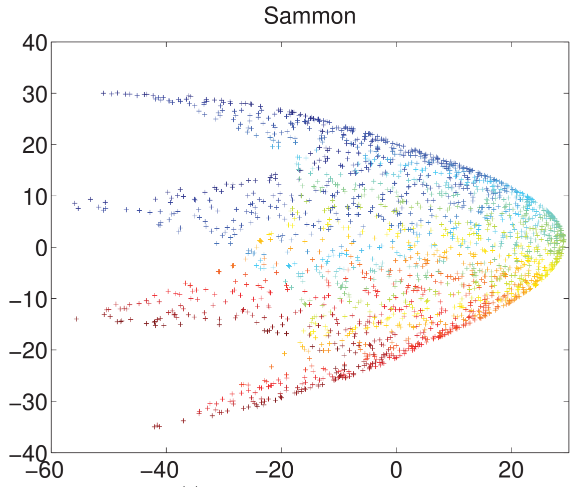
(c) LLE



(d) Laplacian Eigenmap, 2nd and 3rd

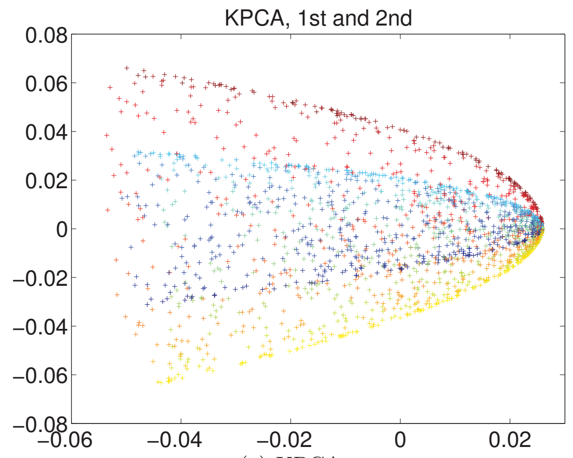


(e) Standard PCA

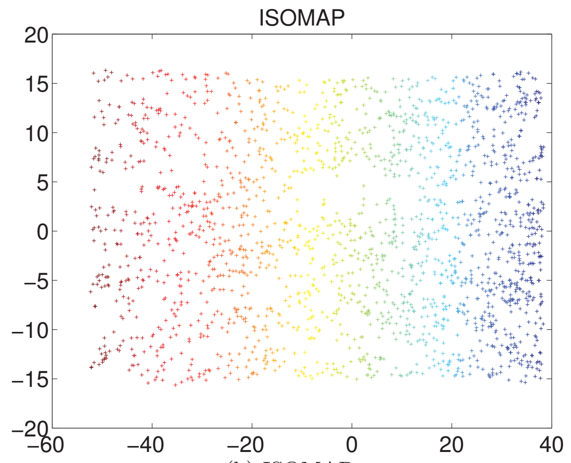


(f) Sammon's mapping

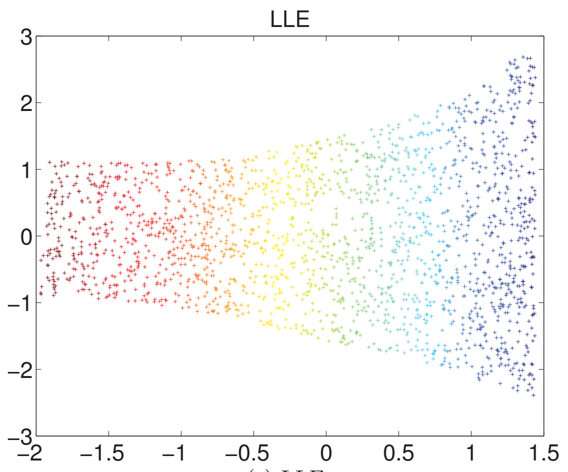
Figure 2.6: Results of nonlinear mapping algorithms on the *parabolic* data set. "2nd and 3rd" in the captions means that we are showing the second and the third components, instead of the first two.



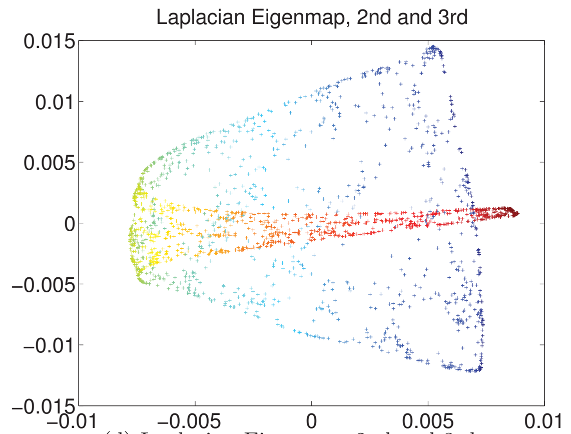
(a) KPCA



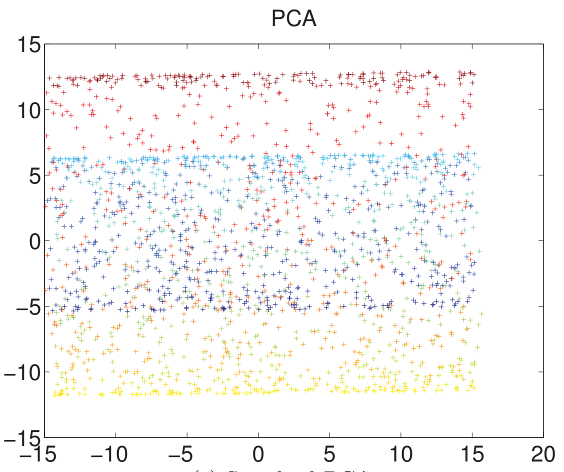
(b) ISOMAP



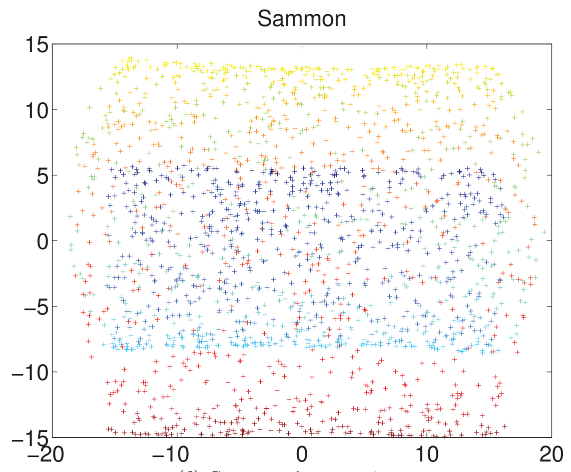
(c) LLE



(d) Laplacian Eigenmap, 2nd and 3rd

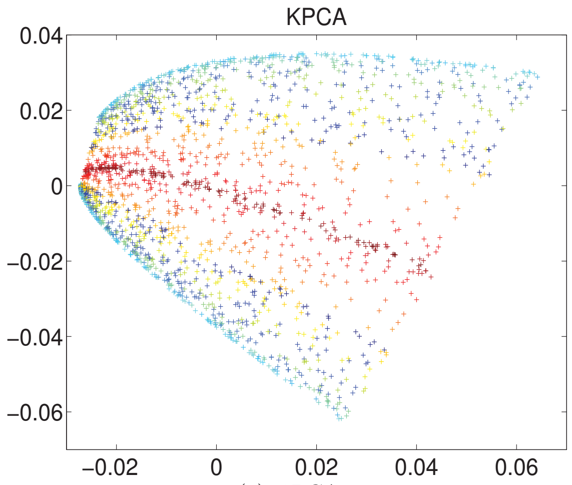


(e) Standard PCA

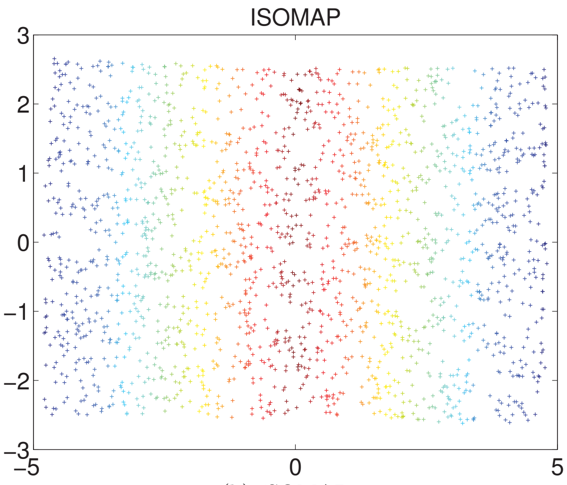


(f) Sammon's mapping

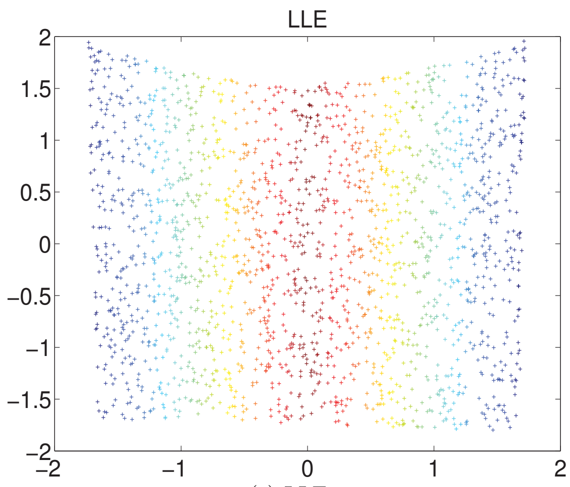
Figure 2.7: Results of nonlinear mapping algorithms on the *swiss roll* data set. "2nd and 3rd" in the captions means that we are showing the second and the third components, instead of the first two.



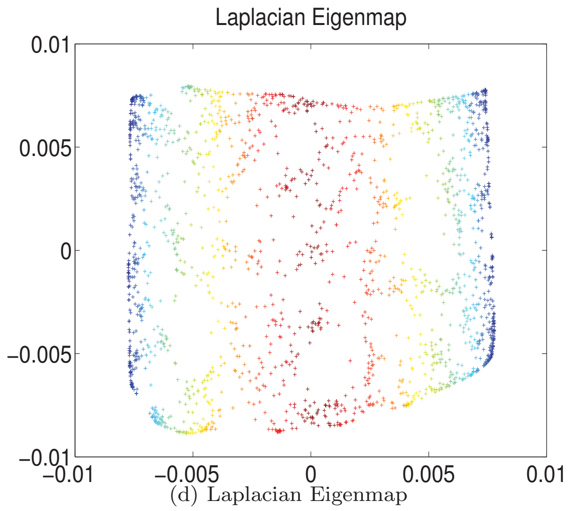
(a) KPCA



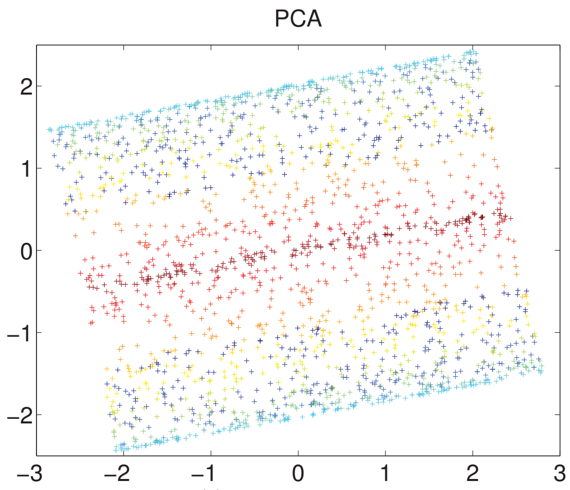
(b) ISOMAP



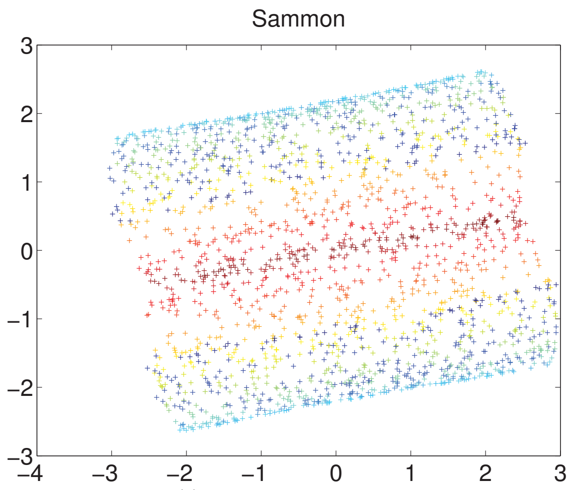
(c) LLE



(d) Laplacian Eigenmap

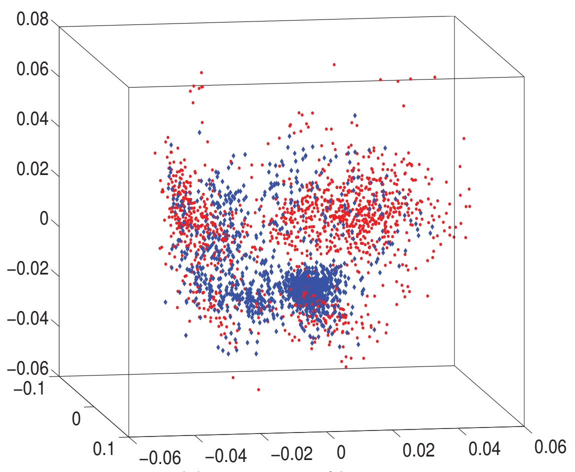


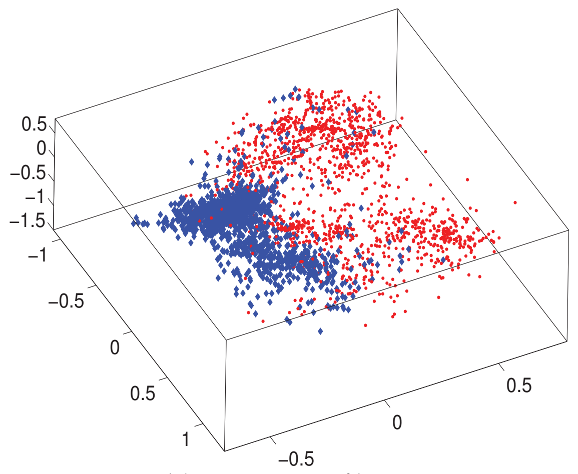
(e) Standard PCA

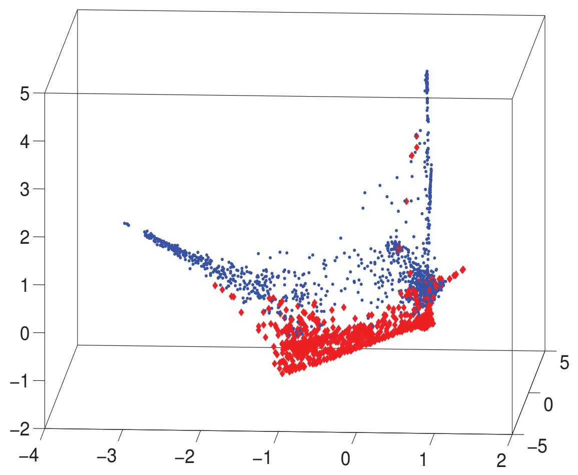


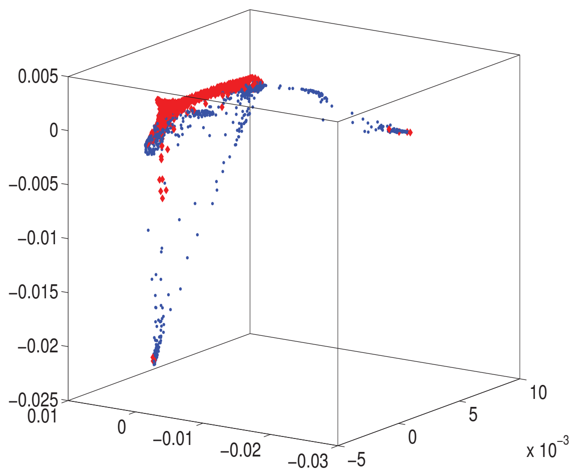
(f) Sammon's mapping

Figure 2.8: Results of nonlinear mapping algorithms on the *S-curve* data set.

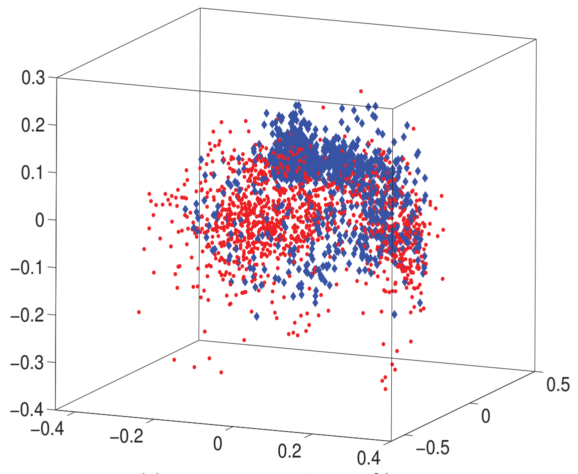

(a) KPCA,35.2% error


(b) ISOMAP, 15.9% error


(c) LLE, 9.2% error



(d) Laplacian Eigenmap, 41.5% error


(e) Standard PCA, 36.4% error

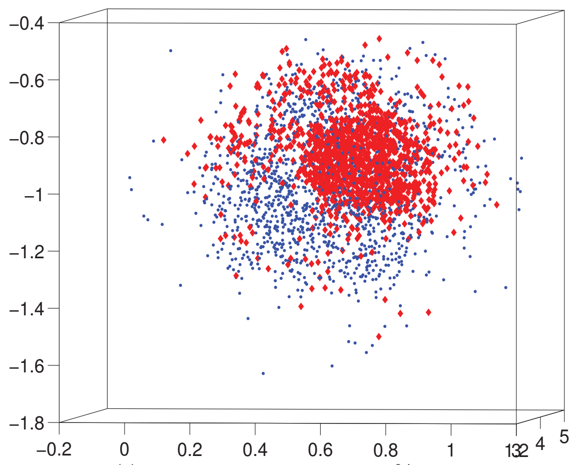

(f) Sammon's mapping, 39.3% error

Figure 2.9: Results of nonlinear mapping algorithms on the face images. The two classes (Asians and non-Asians) are shown in two different colors. The (training) error rates by applying quadratic discriminant analysis on the low dimensional data points are shown in the captions.

Chapter 3

Incremental Nonlinear Dimensionality Reduction By Manifold Learning

In Chapter 2, we discussed different algorithms to achieve dimensionality reduction by nonlinear mapping. Most of these nonlinear mapping algorithms operate in a batch mode[1], meaning that all the data points need to be available during training. In applications like surveillance, where (image) data are collected sequentially, batch method is computationally demanding: repeatedly running the "batch" version whenever new data points become available takes a long time. It is wasteful to discard previous computation results. Data accumulation is particularly beneficial to manifold learning algorithms due to their non-parametric nature. Another reason for developing incremental (non-batch) methods is that the gradual changes in the data manifold can be visualized. As more and more data points are obtained, the evolution of the data manifold can reveal interesting properties of the data stream. Incremental learning can also help us to decide when we should stop collecting data: if there is no noticeable change in the learning result with the additional data collected, there is no point in continuing. The intermediate result produced by an incremental algorithm can prompt us about the existence of any "problematic" region: we can focus the remaining data collection effort on that region. An incremental algorithm can be easily modified to incorporating "forgetting", i.e., the old data points gradually lose their significance. The algorithm can then adjust the manifold in the presence of the drifting of data characteristics. Incremental learning is also useful when there is an unbounded stream of possible data to learn from. This situation can arise when a continuous invariance transformation is applied to a finite set of training data to create additional data to reflect pattern invariance.

In this chapter, we describe a modification of the ISOMAP algorithm so that it can update the low dimensional representation of data points efficiently as additional samples become available. Both the original ISOMAP algorithm [248] and its landmark points version [55] are considered. We are interested in ISOMAP because it is intuitive, well understood, and produces good mapping results [133, 276]. Furthermore, there are theoretical studies supporting the use of ISOMAP, such as its convergence proof [18] and the conditions for successful recovery of co-ordinates [66]. There is also a continuum extension of ISOMAP [282] as well as a spatio-temporal extension [133]. However,

[1]Sammon's mapping can be implemented by a feed-forward neural network [180] and hence can be made online if an online training rule is used.

the motivation of our work is applicable to other mapping algorithms as well.

The main contributions of this chapter include:

1. An algorithm that efficiently updates the solution of the all-pairs shortest path problems. This contrasts with previous work like [193], where different shortest path trees are updated independently.

2. More accurate mappings for new points by a superior estimate of the inner products.

3. An incremental eigen-decomposition problem with increasing matrix size is solved by subspace iteration with Ritz acceleration. This differs from previous work [270] where the matrix size is assumed to be constant.

4. A vertex contraction procedure that improves the geodesic distance estimate without additional memory.

The rest of this chapter is organized as follows. After a recap of ISOMAP in section 3.1, the proposed incremental methods are described in section 3.2. Experimental results are presented in section 3.3, followed by discussions in section 3.4. Finally, in section 3.5 we conclude and describe some topics for future work.

3.1 Details of ISOMAP

The basic idea of the ISOMAP algorithm was presented in Section 2.6. It maps a high dimensional data set $\mathbf{y}_1, \ldots, \mathbf{y}_n$ in $\mathbb{R}^D$ to its low dimensional counterpart $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbb{R}^d$, in such a way that the *geodesic distance* between $\mathbf{y}_i$ and $\mathbf{y}_j$ on the data manifold is as close to the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ in $\mathbb{R}^d$ as possible. In this section, we provide more algorithmic details on how the mapping is done. This also defines the notation that we are going to use throughout this chapter.

The ISOMAP algorithm has three stages. First, a neighborhood graph is constructed. Let Δ_{ij} be the (Euclidean) distance between $\mathbf{y}_i$ and $\mathbf{y}_j$. A weighted undirected neighborhood graph $\mathcal{G} = (V, E)$ with the vertex $v_i \in V$ corresponding to $\mathbf{y}_i$ is constructed. An edge $e(i, j)$ between v_i and v_j exists if and only if $\mathbf{y}_i$ is a neighbor of $\mathbf{y}_j$, i.e., $\mathbf{y}_i \in N(\mathbf{y}_j)$. The weight of $e(i, j)$, denoted by w_{ij}, is set to Δ_{ij}. The set of indices of the vertices adjacent to v_i in $\mathcal{G}$ is denoted by $adj(i)$.

ISOMAP proceeds with the estimation of geodesic distances. Let g_{ij} denote the length of the shortest path $sp(i, j)$ between v_i and v_j. The shortest paths are found by the Dijkstra's algorithm with different source vertices. The shortest paths can be stored efficiently by the predecessor matrix π_{ij}, where $\pi_{ij} = k$ if v_k is immediately before v_j in $sp(i, j)$. If there is no path from v_i to v_j, π_{ij} is set to 0. Conceptually, however, it is useful to imagine a shortest path tree $\mathcal{T}(i)$, where the root node is v_i and $sp(i, j)$ consists of the tree edges from v_i to v_j. The subtree of $\mathcal{T}(i)$ rooted at v_a is denoted by $\mathcal{T}(i; a)$. Since g_{ij} is the approximate geodesic distance between $\mathbf{y}_i$ and $\mathbf{y}_j$, we shall call g_{ij} the "geodesic distance". Note that $\mathbf{G} = \{g_{ij}\}$ is a symmetric matrix.

Finally, ISOMAP recovers $\mathbf{x}_i$ by using the classical scaling [49] on the geodesic distance. Define $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$. Compute $\mathbf{B} = -1/2\mathbf{H}\tilde{\mathbf{G}}\mathbf{H}$, where $\mathbf{H} = \{h_{ij}\}$, $h_{ij} = \delta_{ij} - 1/n$ and δ_{ij} is the delta function, i.e., $\delta_{ij} = 1$ if $i = j$ and 0 otherwise. The entries $\tilde{g}_{ij}$ of $\tilde{\mathbf{G}}$ are simply g_{ij}^2. We seek $\mathbf{X}^T\mathbf{X}$ to be as close to $\mathbf{B}$ as possible in the least square sense. This is done by setting $\mathbf{X} = [\sqrt{\lambda_1}\mathbf{v}_1 \ \ldots \sqrt{\lambda_d}\mathbf{v}_d]^T$, where $\lambda_1, \ldots, \lambda_d$ are the d largest eigenvalues of $\mathbf{B}$, with corresponding eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_d$.

3.2 Incremental Version of ISOMAP

The key computation in ISOMAP involves solving an all-pairs shortest path problem and an eigen-decomposition problem. As new data arrive, these quantities usually do not change much: a new vertex in the graph often changes the shortest paths among only a subset of the vertices, and the simple eigenvectors and eigenvalues of a slightly perturbed real symmetric matrix stay close to their original values. This justifies the reuse of the current geodesic distance and co-ordinate estimates for update. We restrict our attention to $k$nn neighborhood, since ϵ-neighborhood is awkward for incremental learning: the neighborhood size should be constantly decreasing as additional data points become available.

The problem of incremental ISOMAP can be stated as follows. Assume that the low dimensional co-ordinates $\mathbf{x}_i$ of $\mathbf{y}_i$ for the first n points are given. We observe the new sample $\mathbf{y}_{n+1}$. How should we update the existing set of $\mathbf{x}_i$ and find $\mathbf{x}_{n+1}$? Our solution consists of three stages. The geodesic distances g_{ij} are first updated in view of the change of neighborhood graph due to v_{n+1}. The geodesic distances of the new point to the existing points are then used to estimate $\mathbf{x}_{n+1}$. Finally, all $\mathbf{x}_i$ are updated in view of the change in g_{ij}.

In section 3.2.1, we shall describe the modification of the original ISOMAP for incremental updates. A variant of ISOMAP that utilizes the geodesic distances from a fixed set of points (landmark points) [55] is modified to become incremental in section 3.2.2. Because ISOMAP is non-parametric, the data points themselves need to be stored. Section 3.2.3 describes a vertex contraction procedure, which improves the geodesic distance estimate with the arrival of new data without storing the new data. This procedure can be applied to both the variants of ISOMAP. Throughout this section we assume d (dimensionality of the projected space) is fixed. This can be estimated by analyzing either the spectrum of the target inner product matrix or the residue of the low rank approximation as in [248], or by other methods to estimate the intrinsic dimensionality of a manifold [143, 171, 47, 35, 33, 259, 207].

3.2.1 Incremental ISOMAP: Basic Version

We shall modify the original ISOMAP algorithm [248] (summarized in section 3.1) to become incremental. Details of the algorithms as well as an analysis of their time complexity are given in Appendix A. Throughout this section, the shortest paths are represented by the more economical predecessor matrix, instead of multiple shortest path trees $\mathcal{T}(i)$.

3.2.1.1 Updating the Neighborhood Graph

Let $\mathcal{A}$ and $\mathcal{D}$ denote the set of edges to be added and deleted after inserting v_{n+1} to the neighborhood graph, respectively. An edge $e(i, n+1)$ should be added if (i) v_i is one of the k nearest neighbors of v_{n+1}, or (ii) v_{n+1} replaces an existing vertex and becomes one of the k nearest neighbors of v_i. In other words,

$$\mathcal{A} = \{e(i, n+1) : \Delta_{n+1,i} \leq \Delta_{n+1,\tau_{n+1}} \text{ or } \Delta_{i,n+1} \leq \Delta_{i,\tau_i}\}, \tag{3.1}$$

where τ_i is the index of the k-th nearest neighbor of v_i.

For $\mathcal{D}$, note that a necessary condition to delete the edge $e(i,j)$ is that v_{n+1} replaces v_i (v_j) as one of the k nearest neighbors of v_j (v_i). So, all the edges to be deleted must be in the form $e(i, \tau_i)$ with $\Delta_{i,n+1} \leq \Delta_{i,\tau_i}$. The deletion should proceed if v_i is not one of the k nearest neighbors of v_{τ_i} after inserting v_{n+1}. Therefore,

$$\mathcal{D} = \{e(i, \tau_i) : \Delta_{i,\tau_i} > \Delta_{i,n+1} \text{ and } \Delta_{\tau_i,i} > \Delta_{\tau_i,\iota_i}\}, \tag{3.2}$$

```
1: Input: $e(a, b)$, the edge to be removed; $\{g_{ij}\}$; $\{\pi_{ij}\}$
2: Output: $F_{(a,b)}$, set of "affected" vertex pairs
3: $R_{ab} := \emptyset; Q$.enqueue($a$);
4: while $Q$.notEmpty do
5:     $t := Q$.pop; $R_{ab} = R_{ab} \cup \{t\}$;
6:     for all $u \in adj(t)$ do
7:         If $\pi_{ub} = a$, enqueue $u$ to $Q$;
8:     end for
9: end while{Construction of $R_{ab}$ finishes when the loop ends.}
10: $F_{(a,b)} := \emptyset$;
11: Initialize $\mathcal{T}'$, the expanded part of $\mathcal{T}(a; b)$, to contain $v_b$ only;
12: for all $u \in R_{ab}$ do
13:     $Q$.enqueue($b$)
14:     while $Q$.notEmpty do
15:         $t := Q$.pop;
16:         if $\pi_{at} = \pi_{ut}$ then
17:             $F_{(a,b)} = F_{(a,b)} \cup \{(u, t)\}$;
18:             if $v_t$ is a leaf node in $\mathcal{T}'$ then
19:                 for all $v_s$ in $adj(t)$ do
20:                     Insert $v_s$ as a child of $v_t$ in $\mathcal{T}'$ if $\pi_{as} = t$
21:                 end for
22:             end if
23:             Insert all the children of $v_t$ in $\mathcal{T}'$ to the queue $Q$;
24:         end if
25:     end while
26: end for{$\forall\ u \in R_{ab}, \forall\ s \in \mathcal{T}(u; b)$, $sp(u, s)$ uses $e(a, b)$.}
```
Algorithm 3.1: *ConstructFab*: $F_{(a,b)}$, the set of vertex pairs whose shortest paths are invalidated when $e(a, b)$ is deleted, is constructed. R_{ab} is the set of vertices such that if $u \in R_{ab}$, the shortest path between a and u contains $e(a, b)$.

where ι_i is the index of the k-th nearest neighbor of v_{τ_i} after inserting v_{n+1} in the graph. Note that we have assumed there is no tie in the distances. If there are ties, random perturbation can be applied to break the ties.

3.2.1.2 Updating the Geodesic Distances

The deleted edges can break existing shortest paths, while the added edges can create improved shortest paths. This is much more involved than it appears, because the change of a single edge can modify the shortest paths among multiple vertices.

Consider $e(a, b) \in \mathcal{D}$. If $sp(a, b)$ is not simply $e(a, b)$, deletion of $e(a, b)$ has no effect on the geodesic distances. Hence, we shall suppose that $sp(a, b)$ consists of the single edge $e(a, b)$. We propagate the effect of the removal of $e(a, b)$ to the set of vertices R_{ab} (Figure 3.1). R_{ab} is used in turn to construct $F_{(a,b)}$, the set of all (i, j) pairs with $e(a, b)$ in $sp(i, j)$. This is done by *ConstructFab* (Algorithm 3.1), which finds all the vertices v_t under $\mathcal{T}(a; b)$ such that $sp(u, t)$ contains v_b, where $u \in R_{ab}$. The set of vertex pairs whose shortest paths are invalidated due to the removal of edges in $\mathcal{D}$ is thus $F = \cup_{e(a,b) \in D} F_{(a,b)}$. The shortest path distances between these vertex pairs are updated by *ModifiedDijkstra* (Algorithm 3.2) with source vertex v_u and destination vertices $C(u)$. It is similar to the Dijkstra's algorithm, except that only the geodesic distances from v_u to $C(u)$ (instead of all the vertices) are unknown. Note that both v_u and $C(u)$ are derived from F.

The order of the source vertex in invoking *ModifiedDijkstra* can impact the run time significantly. An approximately optimal order is found by interpreting F as an auxiliary graph $\mathcal{B}$ (the undirected edge $e(i, j)$ is in $\mathcal{B}$ iff $(i, j) \in F$), and removing the vertices in $\mathcal{B}$ with the smallest degree in a greedy

```
1: Input: $u$; $C(u)$; $\{g_{ij}\}$; $\{w_{ij}\}$
2: Output: the updated geodesic distances $\{g_{uv}\}$
3: for all $j \in C(u)$ do
4:     $H := adj(j) \cap \big(V/C(u)\big)$;
5:     $\delta(j) = \min_{k \in H}(g_{uk} + w_{kj})$, or $\infty$ if $H = \emptyset$;
6:     Insert $\delta(j)$ to a heap with index $j$;
7: end for
8: while the heap is not empty do
9:     $k :=$ the index of the entry by "Extract Min" on the heap;
10:    $C(u) := C(u)/\{k\}; g_{uk} := \delta(k); g_{ku} := \delta(k)$;
11:    for all $j \in adj(k) \cap C(u)$ do
12:        $dist := g_{uk} + w_{kj}$;
13:        If $g_{uk} + w_{kj} < \delta(j)$, perform "Decrease Key" on $\delta(j)$ to become $dist$;
14:    end for
15: end while
```
Algorithm 3.2: *ModifiedDijkstra*: The geodesic distances from the source vertex u to the set of vertices $C(u)$ are updated.

```
1: Input: Auxiliary graph $\mathcal{B}$
2: Output: None. The geodesic distances are updated as a side-effect
3: $l[i] :=$ an empty linked list, for $i = 1, \ldots, n$;
4: for all $v_u \in \mathcal{B}$ do
5:     $f :=$ degree of $v_u$ in $\mathcal{B}$. Insert $v_u$ to $l[f]$;
6: end for
7: $pos := 1$;
8: for $i := 1$ to $n$ do
9:     If $l[pos]$ is empty, increment $pos$ one by one and until $l[pos]$ is not empty;
10:    Remove $v_u$, a vertex in $l[pos]$, from the graph $\mathcal{B}$;
11:    Call ModifiedDijkstra($u$, $adj(u)$ in $\mathcal{B}$);
12:    for all $v_j$ that is a neighbor of $v_u$ in $\mathcal{B}$ do
13:        Find $f$ such that $v_j \in l[f]$ by an indexing array;
14:        Remove $v_j$ from $l[f]$ if $f = 1$, and move $v_j$ from $l[f]$ to $l[f-1]$ otherwise;
15:        $pos = \min(pos, f - 1)$;
16:    end for
17: end for
```
Algorithm 3.3: *OptimalOrder*: a greedy algorithm to remove the vertex with the smallest degree in the auxiliary graph $\mathcal{B}$. The removal of v_u corresponds to the execution of *ModifiedDijsktra* (Algorithm 3.2) with u as the source vertex.

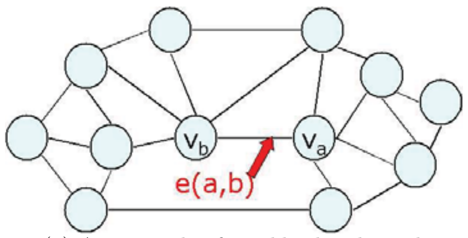
manner (*OptimalOrder*, Algorithm 3.3). When v_u is removed from $\mathcal{B}$, *ModifiedDijkstra* is called with source vertex v_u and $C(u)$ as the neighbors of v_u in $\mathcal{B}$.

The next stage of the algorithm finds the geodesic distances between v_{n+1} and the other vertices. Since all the edges in $\mathcal{A}$ (edges to be inserted) are incident on v_{n+1}, we have

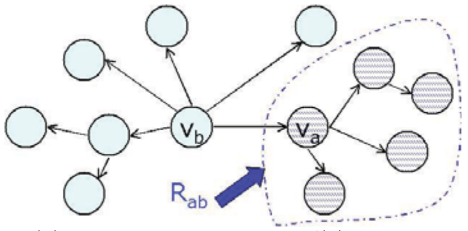
$$g_{n+1,i} = g_{i,n+1} = \min_{\substack{j \text{ such that} \\ e(n+1,j) \in A}} \left(g_{ij} + w_{j,n+1}\right) \quad \forall i. \tag{3.3}$$

Finally, we consider how $\mathcal{A}$ can shorten other geodesic distances. This is done by first locating all the vertex pairs (v_a, v_b), both adjacent to v_{n+1}, such that $v_b \to v_{n+1} \to v_a$ is a better shortest path between v_a and v_b. Starting from v_a and v_b, *UpdateInsert* (Algorithm 3.4) searches for all the vertex pairs that can use the new edge for a better shortest path, based on the updated graph.

For all the priority queues in this section, binary heap is used instead of the asymptotically faster Fibonacci's heap. Since the size of our heap is typically small, binary heap, with a smaller time

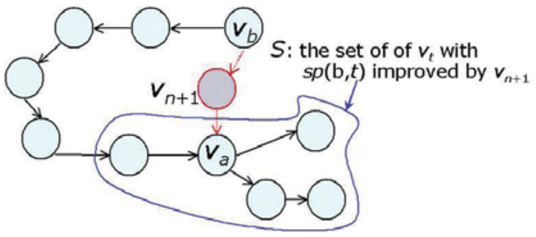


(a) An example of neighborhood graph



(b) The shortest path tree $\mathcal{T}(a)$ and R_{ab}

Figure 3.1: The edge $e(a,b)$ is to be deleted from the neighborhood graph shown in (a). The shortest path tree $\mathcal{T}(a)$ is shown as directed arrows in (b). R_{ab} (c.f. Algorithm 3.1) consists of all the vertices v_u such that $sp(b,u)$ contains $e(a,b)$, i.e., $\pi_{ub} = a$.



v_b
S: the set of of v_t with $sp(b,t)$ improved by v_{n+1}
v_{n+1}
v_a

Figure 3.2: Effect of edge insertion. $\mathcal{T}(a)$ before the insertion of v_{n+1} is represented by the arrows between vertices. The introduction of v_{n+1} creates a better path between v_a and v_b. S denotes the set of vertices such that $t \in S$ iff $sp(b,t)$ is improved by v_{n+1}. Note that v_t must be in $\mathcal{T}(n+1;a)$. For each $u \in S$, *UpdateInsert* (Algorithm 3.4) finds t such that $sp(u,t)$ is improved by v_{n+1}, starting with $t = b$.

constant, is likely to be more efficient.

3.2.1.3 Finding the Co-ordinates of the New Sample

The co-ordinate $\mathbf{x}_{n+1}$ is found by matching its inner product with $\mathbf{x}_i$ to the values derived from the geodesic distances. This approach is in the same spirit as the classical scaling [49] used in ISOMAP. Define $\tilde{\gamma}_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||^2 = ||\mathbf{x}_i||^2 + ||\mathbf{x}_j||^2 - 2\mathbf{x}_i^T\mathbf{x}_j$. Since $\sum_{i=1}^n \mathbf{x}_i = \mathbf{0}$, summation over j and then over i for $\tilde{\gamma}_{ij}$ leads to

$$||\mathbf{x}_i||^2 = \frac{1}{n}(\sum_j \tilde{\gamma}_{ij} - \sum_j ||\mathbf{x}_j||^2),$$
$$\sum_j ||\mathbf{x}_j||^2 = \frac{1}{2n}\sum_{ij} \tilde{\gamma}_{ij}.$$

Similarly, if we define $\gamma_i = ||\mathbf{x}_i - \mathbf{x}_{n+1}||^2$, we have

$$||\mathbf{x}_{n+1}||^2 = \frac{1}{n}(\sum_{i=1}^n \gamma_i - \sum_{i=1}^n ||\mathbf{x}_i||^2),$$
$$\mathbf{x}_{n+1}^T\mathbf{x}_i = -\frac{1}{2}(\gamma_i - ||\mathbf{x}_{n+1}||^2 - ||\mathbf{x}_i||^2) \quad \forall i.$$

```
Input: $a$; $b$; $\{g_{ij}\}$; $\{w_{ij}\}$
Output: $\{g_{ij}\}$ are updated because of the new shortest path $v_a \to v_{n+1} \to v_b$.
$S := \emptyset$; $Q$.enqueue($a$);
while $Q$.notEmpty do
    $t := Q$.pop; $S := S \cup \{t\}$;
    for all $v_u$ that are children of $v_t$ in $\mathcal{T}(n+1)$ do
        if $g_{u,n+1} + w_{n+1,b} < g_{u,b}$ then
            $Q$.enqueue($u$);
        end if
    end for
end while{$S$ has been constructed.}
for all $u \in S$ do
    $Q$.enqueue($b$);
    while $Q$.notEmpty do
        $t := Q$.pop; $g_{ut} := g_{tu} := g_{u,n+1} + g_{n+1,t}$;
        for all $v_s$ that are children of $v_t$ in $\mathcal{T}(n+1)$ do
            if $g_{s,n+1} + w_{n+1,a} < g_{s,a}$ then
                $Q$.enqueue($s$);
            end if
        end for
    end while
end for{$\forall\ u \in S$, update $sp(u,t)$ if $v_{n+1}$ helps.}
```

Algorithm 3.4: *UpdateInsert*: given that $v_a \to v_{n+1} \to v_b$ is a better shortest path between v_a and v_b after the insertion of v_{n+1}, its effect is propagated to other vertices.

If we approximate $\tilde{\gamma}_{ij}$ by g_{ij}^2 and γ_i by $g_{i,n+1}^2$, the target inner product f_i between $\mathbf{x}_{n+1}$ and $\mathbf{x}_i$ can be estimated by

$$2f_i \approx \frac{\sum_j g_{ij}^2}{n} - \frac{\sum_{lj} g_{lj}^2}{n^2} + \frac{\sum_l g_{l,n+1}^2}{n} - g_{i,n+1}^2. \tag{3.4}$$

$\mathbf{x}_{n+1}$ is obtained by solving $\mathbf{X}^T\mathbf{x}_{n+1} = \mathbf{f}$ in the least-square sense, where $\mathbf{f} = (f_1, \ldots, f_n)^T$. One way to interpret the least square solution is by noting that $\mathbf{X} = (\sqrt{\lambda_1}\mathbf{v}_1 \ \ldots \sqrt{\lambda_d}\mathbf{v}_d)^T$, where $(\lambda_i, \mathbf{v}_i)$ is an eigenpair of the target inner product matrix. The least square solution can be written as

$$\mathbf{x}_{n+1} = (\frac{1}{\sqrt{\lambda_1}}\mathbf{v}_1^T\mathbf{f}, \ldots, \frac{1}{\sqrt{\lambda_d}}\mathbf{v}_d^T\mathbf{f})^T. \tag{3.5}$$

The same estimate is obtained if Nyström approximation [89] is used.

A similar procedure is used to compute the out-of-sample extension of ISOMAP in [55, 17]. However, there is an important difference: in these studies, the inner product between the new sample and the existing points is estimated by

$$2\tilde{f}_i = \sum_{j=1}^{n} \frac{g_{ij}^2}{n} - g_{i,n+1}^2. \tag{3.6}$$

It is unclear how this estimate is derived. This estimate is different from that in Equation (3.4) because $\sum_l g_{l,n+1}^2/n - \sum_{ij} g_{ij}^2/n^2$ does not vanish in general; in fact, most of the time this is a large number. Empirical comparisons indicate that our inner product estimate given in Equation (3.4) is much more accurate than the one in Equation (3.6).

Finally, the new mean is subtracted from $x_i, i = 1, \ldots, (n+1)$, to ensure $\sum_{i=1}^{n+1} \mathbf{x}_i = 0$, in order to conform to the convention in the standard ISOMAP.

3.2.1.4 Updating the Co-ordinates

The co-ordinates $\mathbf{x}_i$ should be updated in view of the modified geodesic distance matrix $\mathbf{G}_{\text{new}}$. This can be viewed as an incremental eigenvalue problem, as $\mathbf{x}_i$ can be obtained by eigen-decomposition. However, since the size of the geodesic distance matrix is increasing, traditional methods (such as those described in [270] or [30]) cannot be applied directly. We update $\mathbf{X}$ by finding the eigenvalues and eigenvectors of $\mathbf{B}_{\text{new}}$ by an iterative scheme. Note that gradient descent can be used instead [168].

A good initial guess for the subspace of dominant eigenvectors of $\mathbf{B}_{\text{new}}$ is the column space of $\mathbf{X}^T$. Subspace iteration together with Rayleigh-Ritz acceleration [96] is used to find a better eigen-space:

1. Compute $\mathbf{Z} = \mathbf{B}_{\text{new}}\mathbf{V}$ and perform QR decomposition on $\mathbf{Z}$, i.e., we write $\mathbf{Z} = \mathbf{QR}$ and let $\mathbf{V} = \mathbf{Q}$.

2. Form $\mathbf{Z} = \mathbf{V}^T\mathbf{B}_{\text{new}}\mathbf{V}$ and perform eigen-decomposition of the d by d matrix $\mathbf{Z}$. Let λ_i and $\mathbf{u}_i$ be the i-th eigenvalue and the corresponding eigenvector.

3. $\mathbf{V}_{\text{new}} = \mathbf{V}[\mathbf{u}_1 \dots \mathbf{u}_d]$ is the improved set of eigenvectors of $\mathbf{B}_{\text{new}}$.

Since d is small, the time for eigen-decomposition of $\mathbf{Z}$ is negligible. We do not use any variant of inverse iteration because $\mathbf{B}_{\text{new}}$ is not sparse and its inversion takes $O(n^3)$ time.

3.2.1.5 Complexity

In Appendix A.4, we show that the overall complexity of the geodesic distance update can be written as $O(q(|F| + |H|) + \mu\nu \log \nu + |\mathcal{A}|^2)$, where F and H contain vertex pairs whose geodesic distances are lengthened and shortened because of v_{n+1}, respectively, q is the maximum degree of the vertices in the graph, μ is the number of vertices with non-zero degree in $\mathcal{B}$, and $\nu = \max_i \kappa_i$. Here, κ_i is the degree of the i-th vertex removed from the auxiliary graph $\mathcal{B}$ in Algorithm 3.3. We conjecture that ν, on average, is of the order $O(\log \mu)$. Note that $\mu \leq 2|F|$. The complexity is thus $O(q(|F| + |H|) + \mu \log \mu \log \log \mu + |\mathcal{A}|^2)$. In practice, the first two terms dominate, leading to the effective complexity $O(q(|F| + |H|)$.

We also want to point out that Algorithm 3.2 is fairly efficient; its complexity to solve the all-pairs shortest path by updating all geodesic distances is $O(n^2 logn + n^2 q)$. This is the same as the complexity of the best known algorithm for the all-pairs shortest path problem of a sparse graph, which involves running Dijkstra's algorithm multiple times with different source vertices. For the update of co-ordinates, subspace iteration takes $O(n^2)$ time because of the matrix multiplication.

3.2.2 ISOMAP With Landmark Points

One drawback of the original ISOMAP is its quadratic memory requirement: the geodesic distance matrix is dense and is of size $O(n^2)$, making ISOMAP infeasible for large data sets. Landmark ISOMAP was proposed in [55] to reduce the memory requirement while lowering the computation cost. Instead of all the pairwise geodesic distances, landmark ISOMAP finds a mapping that preserves the geodesic distances originating from a small set of "landmark points". This idea is not entirely new, and the authors in [25] refer to it as the "reference point approach" in the context of embedding.

Without loss of generality, let the first m points, i.e., $\mathbf{y}_1, \dots, \mathbf{y}_m$, be the landmark points. After constructing the neighborhood graph as in the original ISOMAP, landmark ISOMAP uses

```
$\mathcal{I} := \emptyset$;
**for all** $(r_i, s_i, w_i^{\text{old}}, w_i^{\text{new}})$ in the input **do**
  Swap $r_i$ and $s_i$ if $v_{r_i}$ is a child of $v_{s_i}$ in $\mathcal{T}(a)$;
  **if** $v_{s_i}$ is a child of $v_{r_i}$ in $\mathcal{T}(a)$ **then**
    $\mathcal{J} := \{v_{s_i}\} \cup$ descendent of $v_{s_i}$ in $\mathcal{T}(a)$;
    $g_{aj} = g_{aj} + w_i^{\text{new}} - w_i^{\text{old}} \quad \forall j \in \mathcal{J}$;
    $\mathcal{I} = \mathcal{I} \cup \mathcal{J}$;
  **end if**
**end for**
**for all** $j \in \mathcal{J}$ **do**
  $b := \min_{k \in adj(j)} g_{ak} + w_{kj}$; {Find a new path to $v_j$}
  $Q$.enqueue($j$, $\arg\min_{k \in adj(j)} g_{ak} + w_{kj}$, $b$) if $b < g_a j$
**end for**
```
Algorithm 3.5: InitializeEdgeWeightIncrease for the shortest path tree from v_a, $\mathcal{T}(a)$. The inputs are the four tuples $(r_i, s_i, w_i^{\text{old}}, w_i^{\text{new}})$, meaning the weight of $e(r_i, r_j)$ should increase from w_i^{old} to w_i^{new}. Q is the queue of vertices to be processed in Algorithm 3.7.

the Dijkstra's algorithm to compute the $m \times n$ landmark geodesic distance matrix $\mathbf{G} = \{g_{ij}\}$, where g_{ij} is the length of the shortest path between v_i (a landmark point) and v_j. In [55] the authors suggest that $\mathbf{X}$ can be found by first embedding the landmark points and then embedding the remaining points with respect to the landmark points. This is similar to the modification of the Sammon's mapping made by Biswas *et al.* in [25] to cope with large data sets. However, our preliminary experiments indicate that this is not very robust, particularly when the number of landmark points is small. Instead, we follow the implementation of landmark ISOMAP[2] and decompose $\mathbf{B} = \mathbf{H}_m \tilde{\mathbf{G}} \mathbf{H}_n$ by singular value decomposition, $\mathbf{B} = \mathbf{U}\mathbf{S}\mathbf{V}^T = (\mathbf{U}(\mathbf{S})^{1/2})(\mathbf{V}(\mathbf{S})^{1/2})^T$, where $\mathbf{U}^T\mathbf{U}$ and $\mathbf{V}^T\mathbf{V}$ are identity matrices of corresponding sizes, and $\mathbf{S}$ is a diagonal matrix of singular values. The vectors corresponding to the largest d singular values are used to construct a low-rank approximation, $\mathbf{B} \approx \mathbf{Q}^T\mathbf{X}$.

3.2.2.1 Incremental Landmark ISOMAP

After updating the neighborhood graph, the incremental version for landmark ISOMAP proceeds with the update of geodesic distances. Since only the shortest paths from a small number of source vertices are maintained, the computation that can be shared among different shortest path trees is limited. Therefore, we update the shortest path trees independently by adopting the algorithm I presented in [193], instead of the algorithm in section 3.2.1.2. First, Algorithm 3.5 is called to initialize the edge weight increase, which includes edge deletion as a special case. Algorithm 3.7 is then executed to rebuild the shortest path tree. Algorithm 3.6 is then called to initialize the edge weight decrease, which includes edge insertion as a special case. Algorithm 3.7 is again called to rebuild the tree. Deletion of edges is done before the addition of edges because this is more efficient in practice.

The co-ordinate of the new point $\mathbf{x}_{n+1}$ is determined by solving a least-square problem similar to that in section 3.2.1.3. The difference is that the columns of $\mathbf{Q}$, instead of $\mathbf{X}$, are used. So, $\mathbf{Q}^T\mathbf{x}_{n+1} = \mathbf{f}$ is solved in the least-square sense. Finally, we use subspace iteration together with Ritz acceleration [236] to improve singular vector estimates. The steps are

1. Perform SVD on the matrix $\mathbf{BX}$, $\mathbf{U}_1\mathbf{S}_1\mathbf{V}_1^T = \mathbf{BX}$

[2]We are referring to the "official" implementation by the authors of ISOMAP in `http://isomap.stanford.edu`.

```
$\mathcal{I} := \emptyset$;
for all $(r_i, s_i, w_i^{\text{old}}, w_i^{\text{new}})$ in the input do
  Swap $r_i$ and $s_i$ if $g_{a,r_i} > g_{a,s_i}$;
  $diff := g_{a,r_i} + w_i^{\text{new}} - g_{a,s_i}$;
  if $diff < 0$ then
    Move $v_{s_i}$ to be a child of $v_{r_i}$ in $\mathcal{T}(a)$;
    $\mathcal{J} := \{v_{s_i}\} \cup$ descendent of $v_{s_i}$ in $\mathcal{T}(a)$;
    $g_{aj} = g_{aj} + diff \qquad \forall j \in \mathcal{J}$;
    $\mathcal{I} = \mathcal{I} \cup \mathcal{J}$;
  end if
end for
for all $j \in \mathcal{J}$ do
  for all $k \in adj(j)$ do
    $Q$.enqueue($k, j, g_{aj} + w_{jk}$) if $g_{aj} + w_{jk} < g_{ak}$
  end for
end for
```

Algorithm 3.6: InitializeEdgeWeightDecrease for the shortest path tree from v_a, $\mathcal{T}(a)$. The inputs are the four tuples $(r_i, s_i, w_i^{\text{old}}, w_i^{\text{new}})$, meaning the weight of $e(r_i, r_j)$ should decrease from w_i^{old} to w_i^{new}. Q is the queue of vertices to be processed in Algorithm 3.7.

```
while $Q$.notEmpty do
  $(i, j, d)$ := 'Extract Min" on $Q$;
  $diff = d - g_{ai}$;
  if $diff < 0$ then
    Move $v_i$ to be a child of $v_j$ in $\mathcal{T}(a)$;
    $g_{ai} = d$;
    for all $k \in adj(i)$ do
      $newd = g_{ai} + w_{ik}$;
      $Q$.enqueue(k,i,$newd$) if $newd < g_{ak}$;
    end for
  end if
end while
```

Algorithm 3.7: Rebuild $\mathcal{T}(a)$ for those vertices in the priority queue Q that need to be updated.

2. Perform SVD on the matrix $\mathbf{B}^T\mathbf{U}_1$, $\mathbf{U}_2\mathbf{S}_2\mathbf{V}_2^T = \mathbf{B}^T\mathbf{U}_1$

3. Set $\mathbf{X}_{\text{new}} = \mathbf{U}_2(\mathbf{S}_2)^{1/2}$ and $\mathbf{Q}_{\text{new}} = \mathbf{U}_1(\mathbf{S}_2)^{1/2}$

As far as time complexity is concerned, the time to update one shortest path tree is $O(\delta_d \log \delta_d + q\delta_d)$, where δ_d is the minimum number of nodes that must change their distance or parent attributes or both [193], and q is the maximum degree of vertices in the neighborhood graph. The complexity of updating the singular vectors is $O(nm)$, which is linear in n, because the number of landmark points m is fixed.

3.2.3 Vertex Contraction

Owing to the non-parametric nature of ISOMAP, the data points collected need to be stored in the memory in order to refine the estimation of the geodesic distances g_{ij} and the co-ordinates $\mathbf{x}_i$. This can be undesirable if we have an arbitrarily large data stream.

One simple solution is to discard the oldest data point when a pre-determined number of data points has been accumulated. This has the additional advantage of making the algorithm adaptive to drifting in data characteristics. The deletion should take place after the completion of all the updates due to the new point. Deleting the vertex v_i is easy: the edge deletion procedure is used to delete all the edges incident on v_i for both ISOMAP and landmark ISOMAP.

We can do better than deletion, however. A vertex contraction heuristic can be used to record the improvement in geodesic distance estimate without storing additional points. Most of the information the new vertex v_{n+1} contains about the geodesic distance estimate is represented by the shortest paths passing through v_{n+1}. Suppose $sp(a,b)$ can be written as $v_a \rightsquigarrow v_i \to v_{n+1} \to v_b$. The geodesic distance between v_a and v_b can be preserved by introducing a new edge $e(i,b)$ with weight $(w_{i,n+1} + w_{n+1,b})$, even though v_{n+1} is deleted. Both the shortest path tree $\mathcal{T}(a)$ and the graph are updated in view of this new edge. This procedure cannot create inconsistency in any shortest path trees, because the subpath of any shortest path is also a shortest path. This heuristic increases the density of the edges in the graph, however.

Which vertex should be contracted? A simple choice is to contract the new vertex v_{n+1} after adjusting for the change of geodesic distances. Alternatively, we can delete the vertices that are most "crowded" so that the points are spread more evenly along the manifold. This can be done by contracting the non-landmark point whose nearest neighbor is the closest to itself.

3.3 Experiments

We have implemented our main algorithm in Matlab, with the graph theoretic parts written in C++. The running time is measured on a Pentium IV 3.2 GHz PC with 512MB memory running Windows XP, using the profiler of Matlab with the java virtual machine turned off.

3.3.1 Incremental ISOMAP: Basic Version

We evaluated the accuracy and the efficiency of our incremental algorithm on several data sets. The first experiment was on the Swiss roll data set. It is a typical benchmark for manifold learning. Because of its "roll" nature, geodesic distances are more appropriate in understanding the structure of this data set than Euclidean distances. Initialization was done by finding the co-ordinate estimate $\mathbf{x}_i$ for 100 randomly selected points using the "batch" ISOMAP, with a $k$nn neighborhood of size 6. Random points from the Swiss roll data set were added one by one, until 1500 points were accumulated. The incremental algorithm described in section 3.2.1 was used to update the co-ordinates. The first two dimensions of $\mathbf{x}_i$ corresponded to the true structure of the manifold. The gap between the second and the third eigenvalues is fairly significant and it is not difficult to determine the intrinsic dimensionality as two for this data set. Figure 3.3 shows several snapshots of the algorithm[3]. The black dots $(\tilde{\mathbf{x}}_i^{(n)})$ and the red circles $(\mathbf{x}_i^{(n)})$ correspond to the co-ordinates estimated by the incremental and the batch version of ISOMAP, respectively. The red circles and the black dots match very well, indicating that the co-ordinates updated by the incremental ISOMAP follow closely with the co-ordinates estimated by the batch version. This closeness can be quantified by an error measure defined as the square root of the mean square error between $\hat{\mathbf{x}}_i^{(n)}$ and $\mathbf{x}_i^{(n)}$, normalized by the total sample variance:

$$\mathcal{E}_n = \sqrt{\frac{\frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i^{(n)} - \hat{\mathbf{x}}_i^{(n)}||^2}{\frac{1}{n}\sum_{i=1}^{n} ||\mathbf{x}_i^{(n)}||^2}}. \tag{3.7}$$

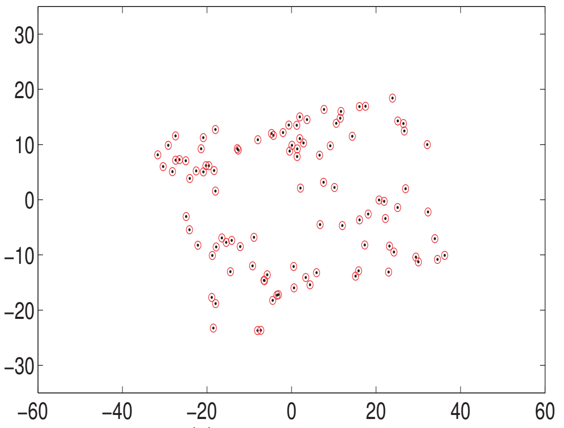
[3]The avi files can be found at `http://www.cse.msu.edu/prip/ResearchProjects/iisomap/`.

Table 3.1: Run time (seconds) for batch and incremental ISOMAP. For batch ISOMAP, computation of $\mathbf{x}_{n+1}$ and updating of $\mathbf{x}_i$ are performed together. Hence there is only one combined run time.

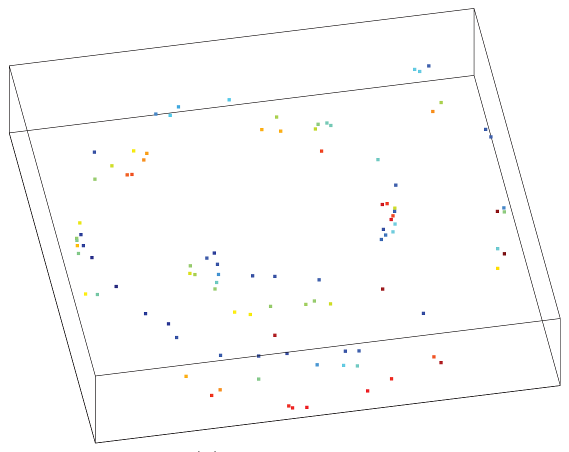
	Swiss roll		S-curve		Rendered face		MNIST 2		`ethn`	
	Batch	Incr.	Batch	Incr.	Batch	Incr.	Batch	Incr.	Batch	Incr.
Neighborhood graph	230.0	1.4	229.9	1.2	20.8	0.4	239.7	1.8	239.3	1.5
Geodesic distance	1618.5	53.2	1632.8	56.2	157.8	5.8	1683.9	41.0	1752.9	39.6
Computing $\mathbf{x}_{n+1}$	804.6	0.9	760.0	0.8	85.1	0.3	671.3	0.8	645.2	1.0
Updating $\mathbf{x}_i$		49.5		49.4		5.5		51.5		48.9

Table 3.2: Run time (seconds) for executing batch and incremental ISOMAP once for different number of points (n). "Dist" corresponds to the time for distance computation for all the n points.

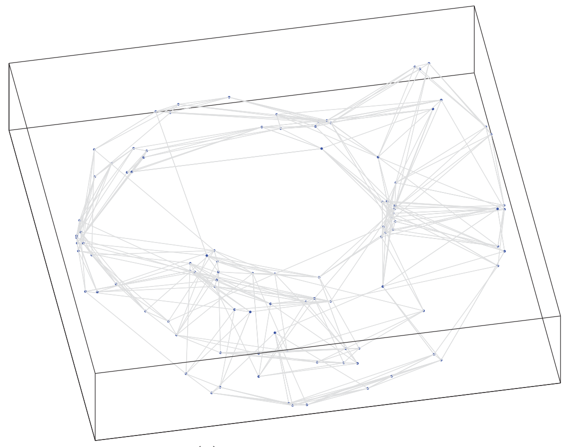
	Swiss roll			S-curve			Rendered face			MNIST 2			`ethn`		
n	Dist.	Batch	Incr.	Dist.	Batch	Incr.	Dist.	Batch	Incr.	Dist.	Batch	Incr.	Dist.	Batch	Incr.
500	0.09	0.66	0.04	0.09	0.64	0.04	0.16	0.60	0.05	0.28	0.62	0.04	1.19	0.61	0.04
1000	0.38	2.62	0.14	0.38	2.47	0.11	N/A	N/A	N/A	1.09	2.34	0.07	4.52	2.45	0.08
1500	0.84	5.72	0.17	0.84	5.65	0.25	N/A	N/A	N/A	2.42	5.41	0.18	10.06	5.65	0.15



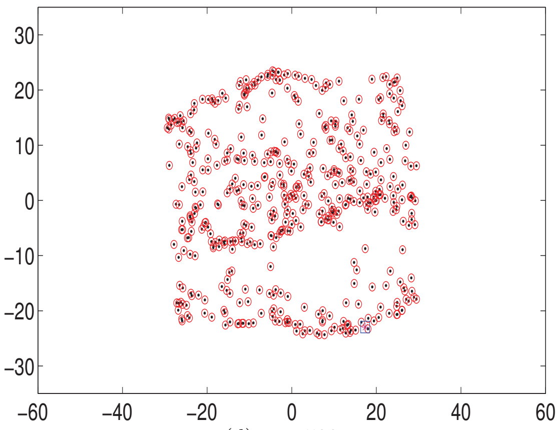
(a) Initial, $n = 100$



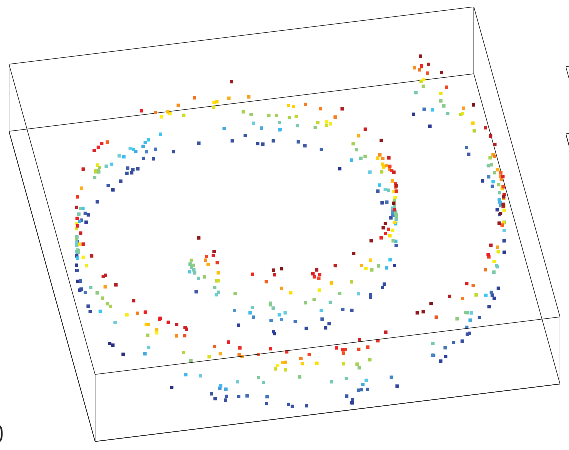
(b) Initial, $n = 100$



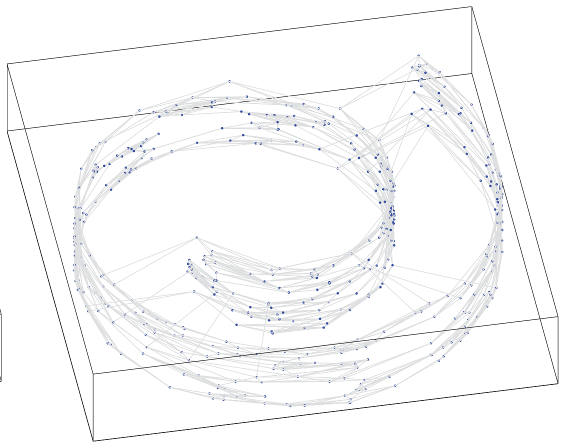
(c) Initial, $n = 100$



(d) $n = 500$

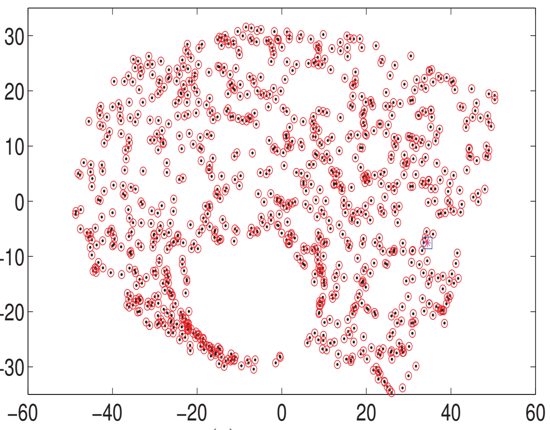


(e) $n = 500$

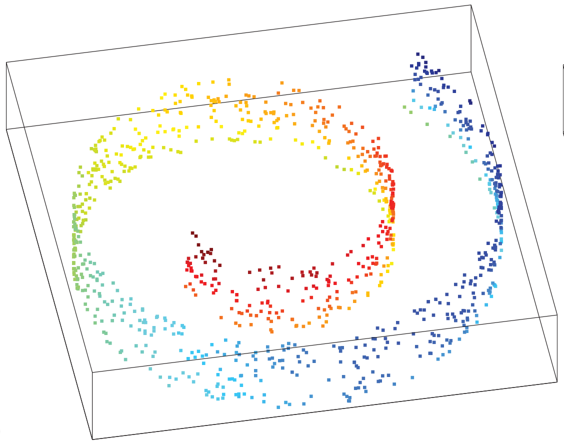


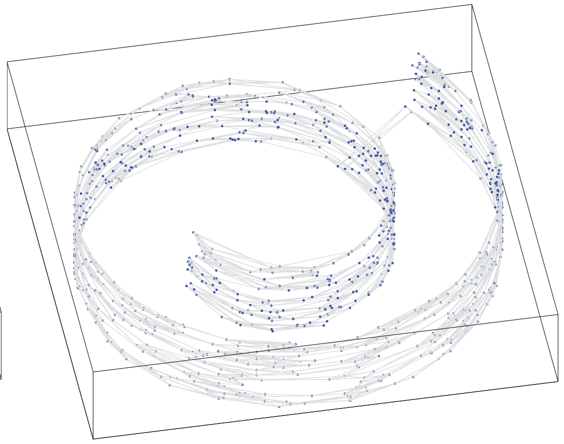
(f) $n = 500$

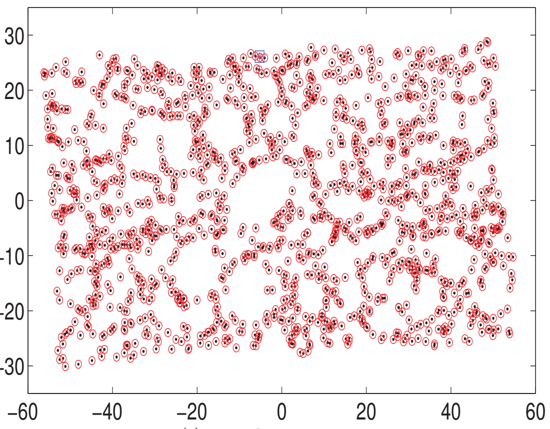
Figure 3.3: Snapshots of "Swiss Roll" for incremental ISOMAP. In the first column, the circles and dots in the figures represent the co-ordinates estimated by the batch and the incremental version, respectively. The square and asterisk denote the co-ordinates of the newly added point, estimated by the batch and the incremental algorithm, respectively. The second column contains scatter plots, where the color of a point corresponds to the value of the most dominant co-ordinate estimated by ISOMAP. The third column illustrates the neighborhood graphs, from which the geodesic distances are estimated.



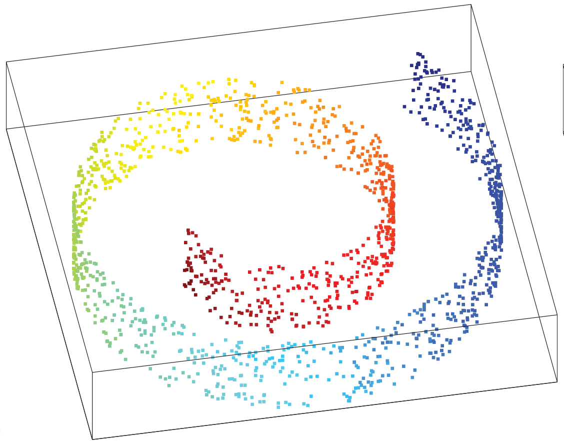
(g) $n = 1000$


(h) $n = 1000$


(i) $n = 1000$



(j) Final, $n = 1500$


(k) Final, $n = 1500$

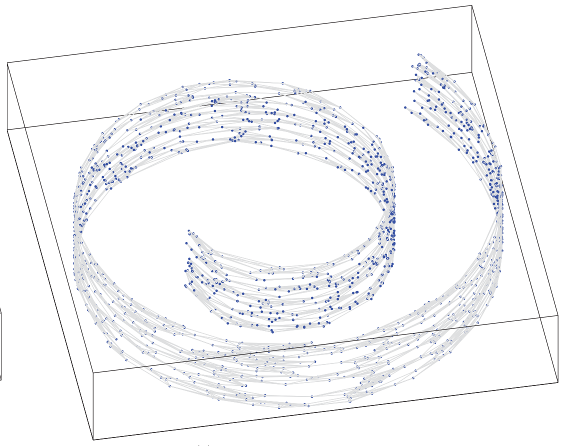
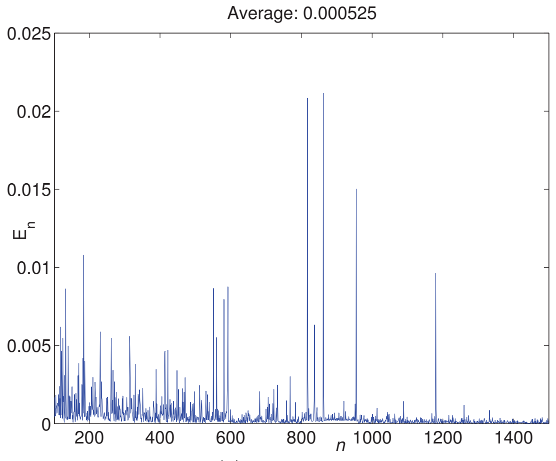
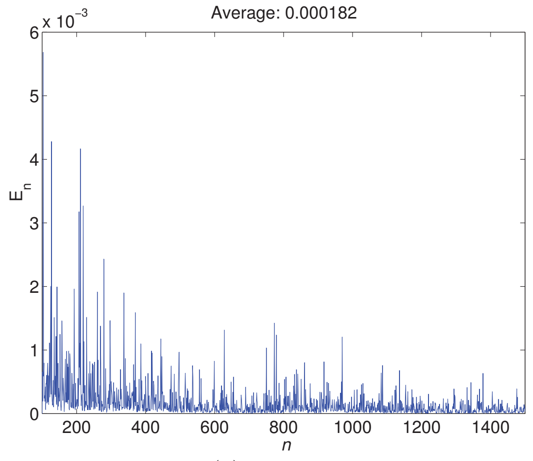

(l) Final, $n = 1500$

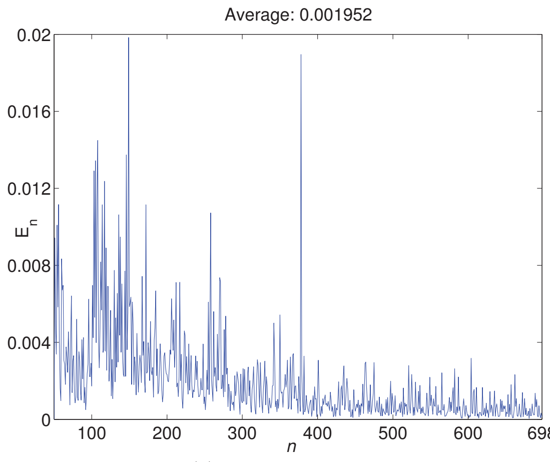
Figure 3.3 (continued)



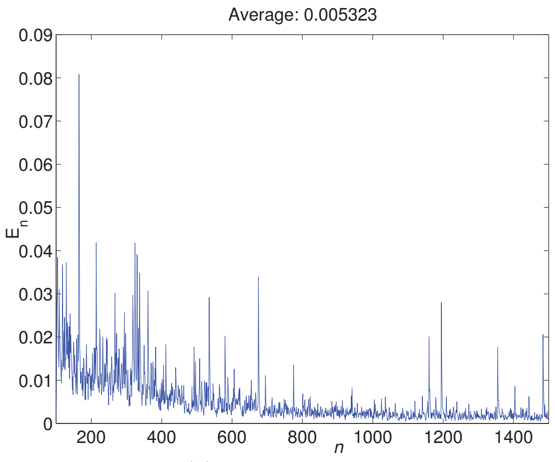
(a) Swiss roll



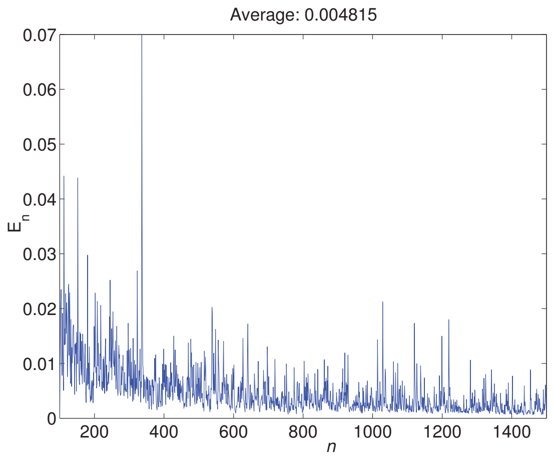
(b) S-curve



(c) Rendered Faces



(d) MNIST digit 2



(e) `ethn`

Figure 3.4: Approximation error ($\mathcal{E}_n$) between the co-ordinates estimated by the basic incremental ISOMAP and the basic batch ISOMAP for different numbers of data points (n) for the five data sets.

Figure 3.4(a) displays $\mathcal{E}_n$ against the number of data points n for Swiss roll. We can see that the proposed updating method is fairly accurate, with an average error of 0.05%. The "spikes" in the graph correspond to the instances where many geodesic distances change dramatically because of the creation or deletion of "short-cuts" in the neighborhood graph. These large errors fade very quickly, however, as evident from the graph.

Table 3.1 shows the computation time decomposed into different tasks. Our incremental approach has significant savings in all three aspects of ISOMAP: graph update, geodesic distance update, and co-ordinate update. The computation time for the distances is not included in the table, because both batch and incremental versions perform the same number of distance computations. Empirically, we observed that for moderate number of data points, the time to update the geodesic distances is longer than the time to update the co-ordinates, whereas the opposite is true when a large number of points have been collected. This is probably due to the fact that the geodesic distances change more rapidly when only a moderate amount of data are collected, whereas the time for matrix multiplication becomes more significant with a larger number of co-ordinates. We have also run the batch algorithm once for different numbers of data points (n). Table 3.2 shows the measured time averaged over 5 identical trials, after excluding the time for distance computation. The time for computing the distances for all the n points, together with the time to run the incremental algorithm once to update when the n-th point arrives, is also included in the table. See Section 3.4 for further discussion of the result.

The co-ordinates estimated with different number of data points are also compared with the co-ordinates estimated with all the available data points. This can give us an additional insight on how the estimated co-ordinates evolve to their final values as new data points gradually arrive. Some snapshots are shown in Figure 3.5.

A similar experimental procedure was applied to other data sets. The "S-curve" data set, another benchmark for manifold learning, contains points in a 3D space lying on a "S"-shaped surface, with an effective dimensionality of two. The "rendered face" data set[4] contains 698 face images with size 64 by 64 rendered at different illumination and pose conditions. Some examples are shown in Figure 3.6. The "MNIST digit 2" data set is derived from the digit images "2" from MNIST[5], and contains 28 by 28 digit images. Several typical images are shown in Figure 3.7. The rendered face data set and the MNIST digit 2 data sets were used in the original ISOMAP paper [248]. Our last data set, `ethn`, contains the face images used in [175]. The task of this data set is to classify a 64 by 64 face image as Asian or non-Asian. This database contains 1320 images for Asian class and 1310 images for non-Asian class, and is composed of several face image databases, including the PF01 database[6], the Yale database[7] the AR database [181], as well as the non-public NLPR database[8]. Some example face images are shown in Figure 3.8. For all these images, the high dimensional feature vectors were created by concatenating the image pixels. The neighborhood size for MNIST digit 2 and `ethn` was set to 10 in order to demonstrate that the proposed approach is efficient and accurate irrespective of the neighborhood used. The approximation error and the computation time for these data sets are shown in Figure 3.4 and Table 3.1. We can see that the incremental ISOMAP is accurate and efficient for updating the co-ordinates for all these data sets.

Since the `ethn` data set is from a supervised classification problem with two classes, we also want to investigate the quality of the ISOMAP mapping with respect to classification. This is

[4]`http://isomap.stanford.edu`
[5]`http://yann.lecun.com/exdb/mnist/`.
[6]`http://nova.postech.ac.kr/archives/imdb.html`.
[7]`http://cvc.yale.edu/projects/yalefaces/yalefaces.html`.
[8]Provided by Dr. Yunhong Wang, National Laboratory for Pattern Recognition, Beijing.

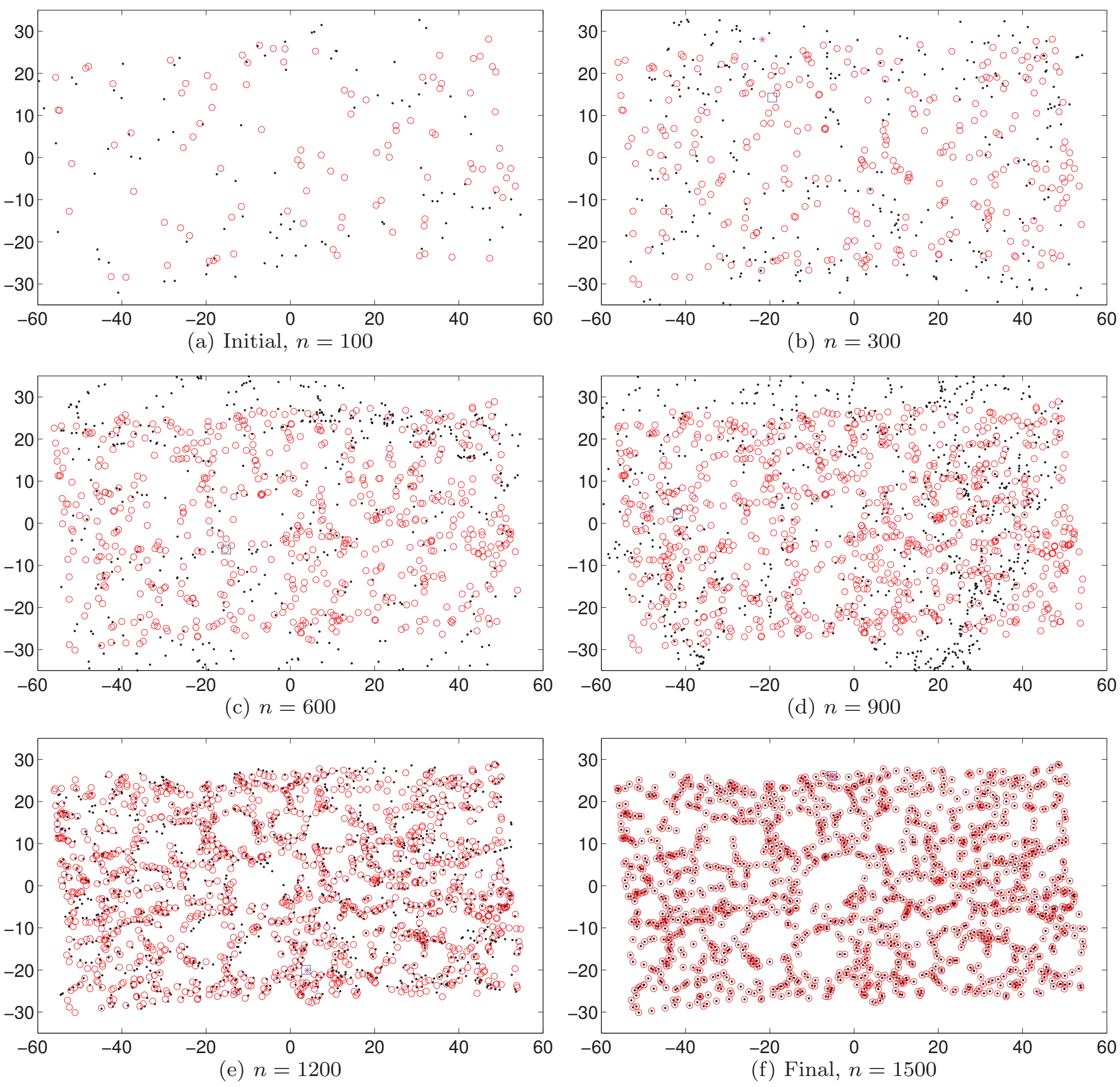

(a) Initial, $n = 100$
(b) $n = 300$
(c) $n = 600$
(d) $n = 900$
(e) $n = 1200$
(f) Final, $n = 1500$

Figure 3.5: Evolution of the estimated co-ordinates for Swiss roll to their final values. The black dots denote the co-ordinates estimated with different number of samples, whereas red circles show the co-ordinates estimated with all the 1500 points. The co-ordinates have been re-scaled to better observe the trend.

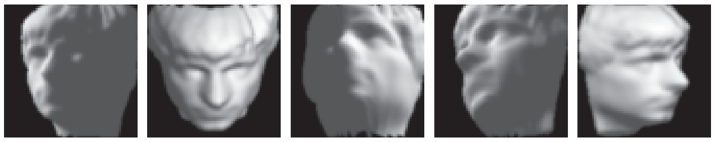


Figure 3.6: Example images from the rendered face image data set. This data set can be found at the ISOMAP web-site.

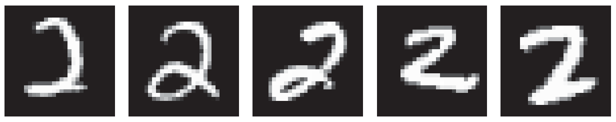


Figure 3.7: Example "2" digits from the MNIST database. The MNIST database can be found at `http://yann.lecun.com/exdb/mnist/`.

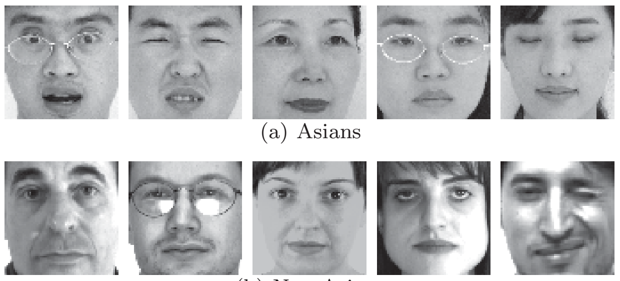


Figure 3.8: Example face images from ethn database.

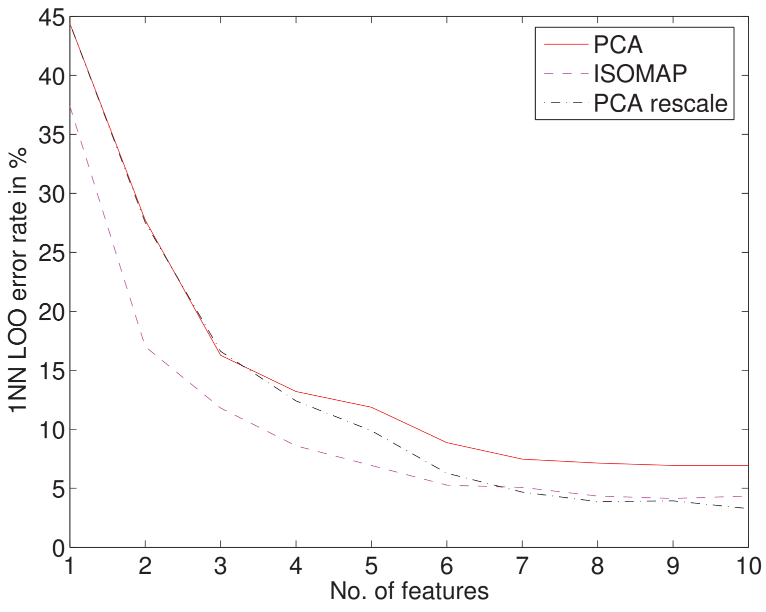


Figure 3.9: Classification performance on ethn database for basic ISOMAP.

done quantitatively by computing the leave-one-out nearest neighbor (with respect to L_2 distance) error rate using different dimensions of the co-ordinates estimated by incremental ISOMAP with 1500 points. For comparison, we project the data linearly to the best hyperplane by PCA and also evaluate the corresponding leave-one-out error rate. Figure 3.9 shows the result. The representation recovered by ISOMAP leads to a smaller error rate than PCA. Note that the performance of PCA can be improved by rescaling each feature so that all of them have equal variance, though the rescaling is essentially a post-processing step, not required by ISOMAP.

3.3.2 Experiments on Landmark ISOMAP

A similar experimental procedure was applied to the incremental landmark ISOMAP described in section 3.2.2 for Swiss roll, S-curve, rendered face, MNIST digit 2, and `ethn` data sets. Starting with 200 randomly selected points from the data set, random points were added until a total of 5000 points[9] accumulated. Forty points from the initial 200 points were chosen randomly to be the landmark points. Snapshots comparing the co-ordinates estimated by the batch version and the incremental version for Swiss roll are shown in Figure 3.10. The approximation error and the computation time are shown in Figure 3.11 and Table 3.3, respectively. The time to run the batch version only once is listed in Table 3.4. Once again, the co-ordinates estimated by the incremental version are accurate with respect to the batch version, and the computation time is much less. We also consider the classification accuracy using landmark ISOMAP on all the 2630 images in the `ethn` data set. The result is shown in Figure 3.12. The co-ordinates estimated by landmark ISOMAP again lead to a smaller error rate than those based on PCA. The difference is more pronounced when the number of dimensions is small (less than five).

3.3.3 Vertex Contraction

The utility of vertex contraction is illustrated in the following experiment. Consider a manifold of a 3-dimensional unit hemisphere embedded in a 10-dimensional space. The geodesic on this manifold is simply the great circle, and the geodesic distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ on the manifold is given by $\cos^{-1}(\mathbf{x}_1^T \mathbf{x}_2)$. Data points lying on this manifold are randomly generated. With $K = 6$, 40 landmark points and 1000 points in memory, vertex contraction is executed until 10000 points are examined. The geodesic distances between the landmark points $\mathcal{X}^L$ and the points in memory $\mathcal{X}^M$ are compared with the ground-truth, and the discrepancy is shown by the solid line in Figure 3.13. As more points are encountered, the error decreases, indicating that vertex contraction indeed improves the geodesic distance estimate. There is, however, a lower limit (around 0.03) on the achievable accuracy, because of the finite size of samples retained in the memory. When additional points are kept in the memory instead of being contracted, the improvement of geodesic distance estimate is significantly slower (the dash-dot line in Figure 3.13). We can see that vertex contraction indeed improves the geodesic distance estimate, partly because it spreads the data points more evenly, and partly because more points are included in the neighborhood effectively.

3.3.4 Incorporating Variance By Incremental Learning

One interesting use of incremental learning is to incorporate invariance by "hallucinating" training data. Given a training sample $\mathbf{y}_i$, additional training data $\mathbf{y}_i^{(1)}, \mathbf{y}_i^{(2)}, \ldots$ can be created by applying different invariance transformations on $\mathbf{y}_i$. The amount of training data can be unbounded, because

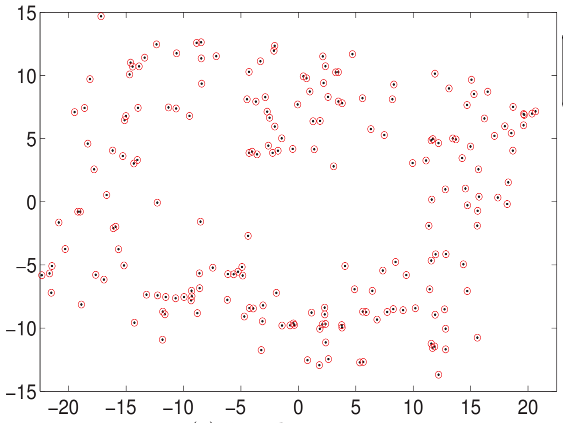
[9]When the data set has less than 5000 points, the experiment stopped after all the points have been used.

Table 3.3: Run time (seconds) for batch and incremental landmark ISOMAP. For batch ISOMAP, computation of $\mathbf{x}_{n+1}$ and updating of $\mathbf{x}_i$ are performed together. Hence there is only one combined run time.

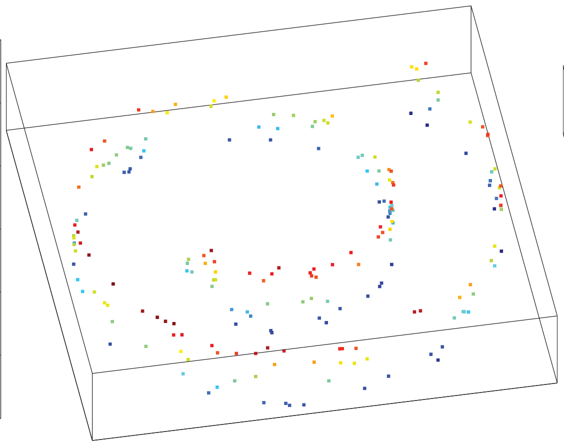
	Swiss roll		S-curve		Rendered face		MNIST 2		`ethn`	
	Batch	Incr.	Batch	Incr.	Batch	Incr.	Batch	Incr.	Batch	Incr.
Neighborhood graph	8745.7	6.6	8692.7	7.1	20.2	0.5	8742.9	7.6	1296.0	3.3
Geodesic distance	824.3	38.9	879.9	39.2	6.5	0.5	913.3	16.9	217.3	6.0
Computing $\mathbf{x}_{n+1}$	199.1	0.5	200.8	0.9	6.6	0.1	210.2	0.9	69.5	0.5
Updating $\mathbf{x}_i$		61.4		62.5		1.2		43.0		17.1

Table 3.4: Run time (seconds) for executing batch and incremental landmark ISOMAP once for different number of points (n). "Dist" corresponds to the time for distance computation for all the n points.

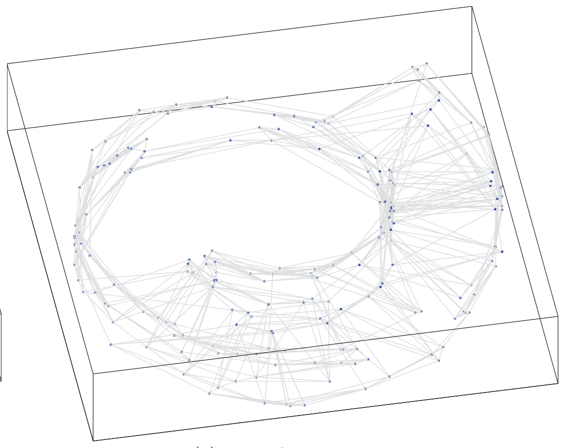
	Swiss roll			S-curve			Rendered face			MNIST 2			`ethn`		
	Dist.	Batch	Incr.	Dist.	Batch	Incr.	Dist.	Batch	Incr.	Dist.	Batch	Incr.	Dist.	Batch	Incr.
500	0.09	0.14	0.01	0.09	0.21	0.03	0.16	0.22	0.02	0.28	0.15	0.02	1.19	0.16	0.02
2000	1.53	1.56	0.02	1.50	1.54	0.03	N/A	N/A	N/A	4.25	1.55	0.01	17.70	1.57	0.02
3500	4.61	11.90	0.03	4.64	12.91	0.06	N/A	N/A	N/A	21.72	7.77	0.04	N/A	N/A	N/A
5000	208.52	168.01	0.05	338.12	174.63	0.06	N/A	N/A	N/A	501.81	226.25	0.05	N/A	N/A	N/A



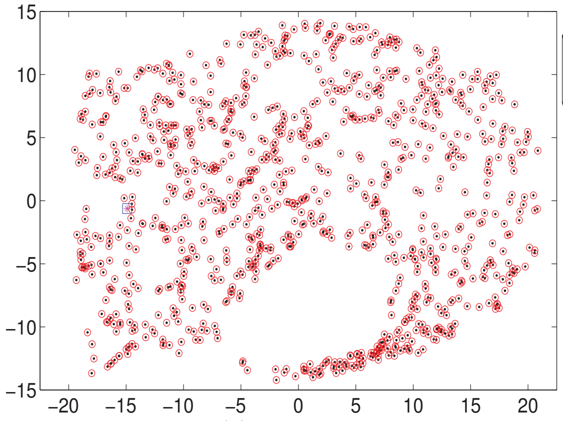
(a) Initial, $n = 200$



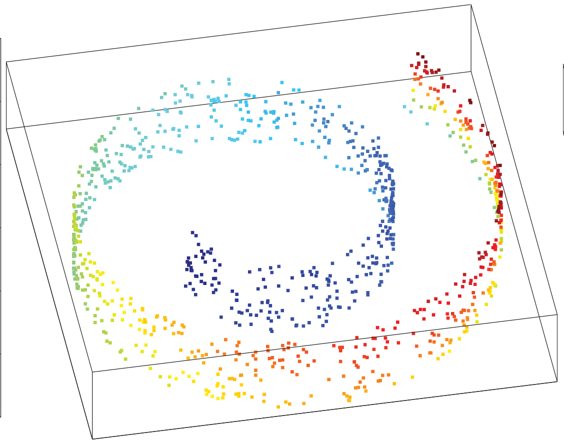
(b) Initial, $n = 200$



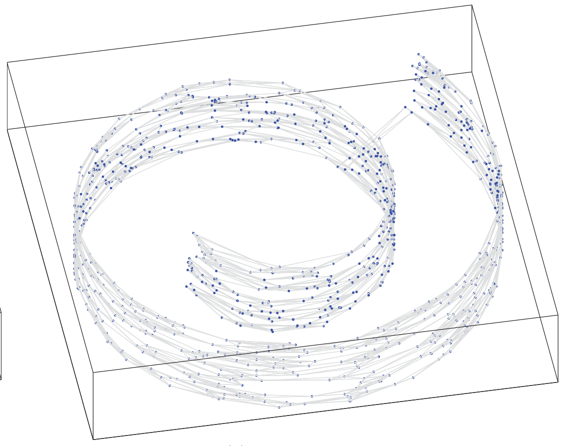
(c) Initial, $n = 200$



(d) $n = 1000$

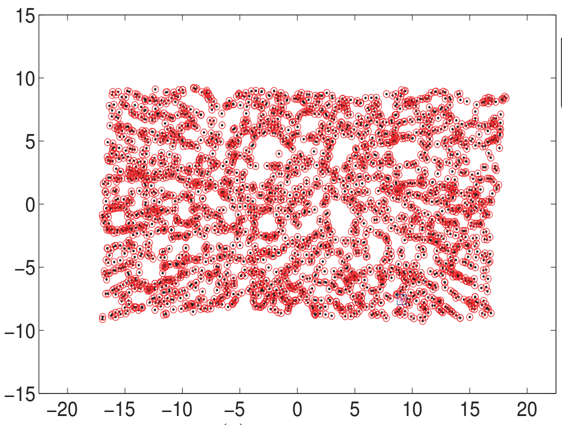


(e) $n = 1000$

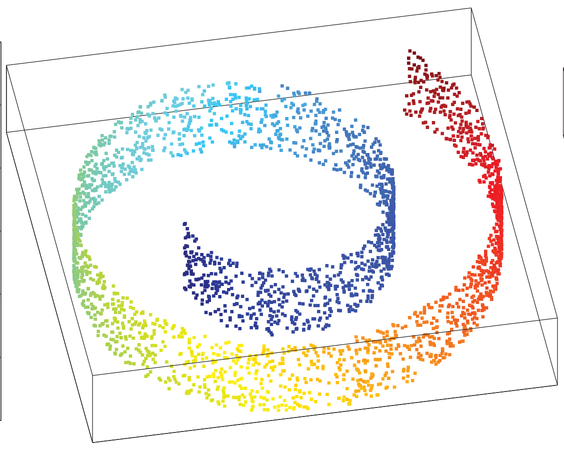


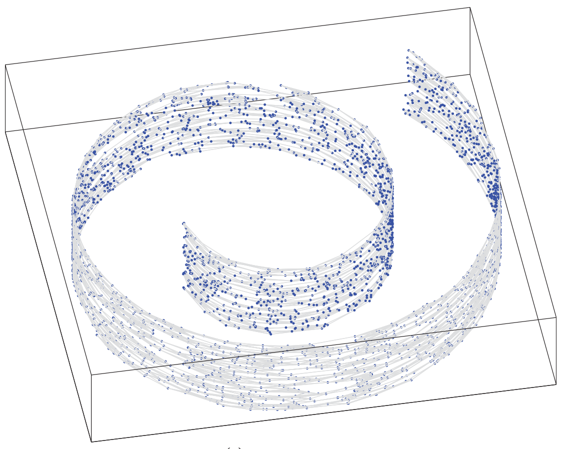
(f) $n = 1000$

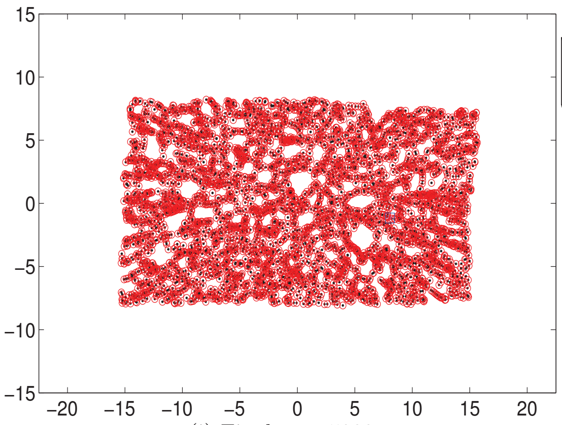
Figure 3.10: Snapshots of "Swiss roll" for incremental landmark ISOMAP. In the first column, the circles and dots in the figures represent the co-ordinates estimated by the batch and the incremental version, respectively. The square and asterisk denote the co-ordinates of the newly added point, estimated by the batch and the incremental algorithm, respectively. The second column contains scatter plots, where the color of a point corresponds to the value of the most dominant co-ordinate estimated by ISOMAP. The third column illustrates the neighborhood graphs, from which the geodesic distances are estimated. It is similar to Figure 3.3, except that the landmark version of ISOMAP is used instead.



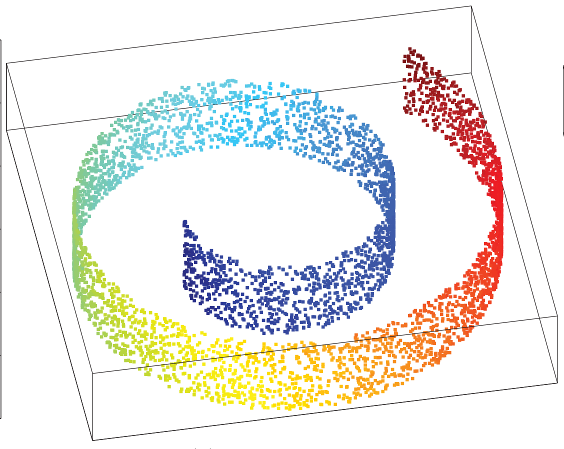
(g) $n = 3000$


(h) $n = 3000$


(i) $n = 3000$



(j) Final, $n = 5000$


(k) Final, $n = 5000$

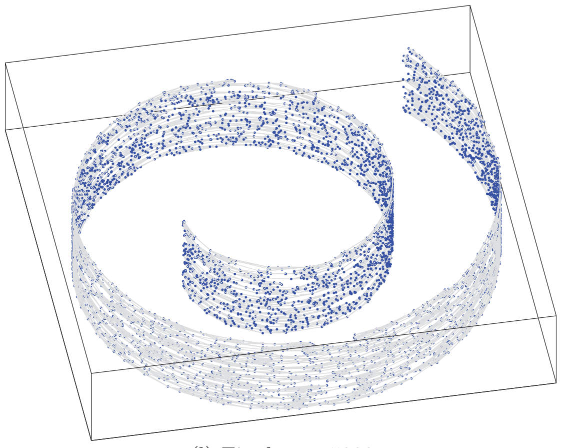
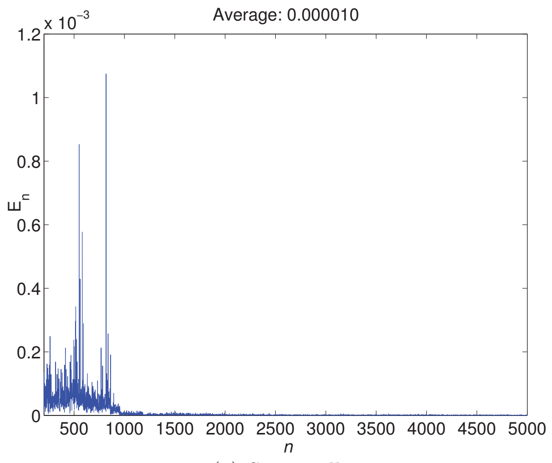
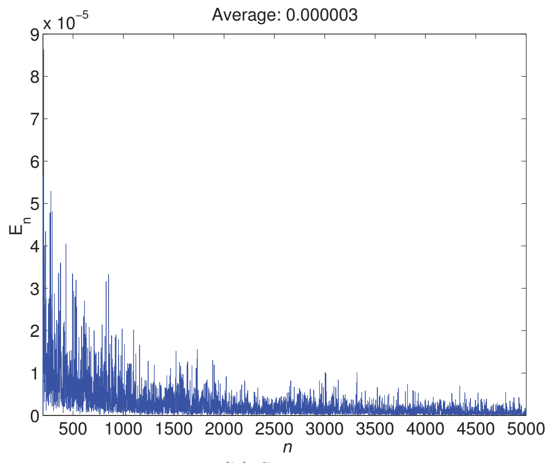

(l) Final, $n = 5000$

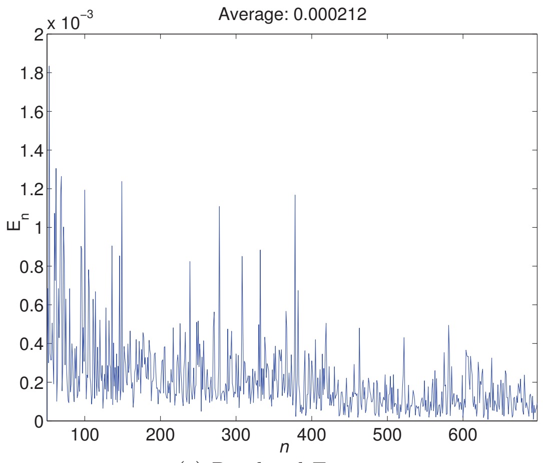
Figure 3.10 (continued)



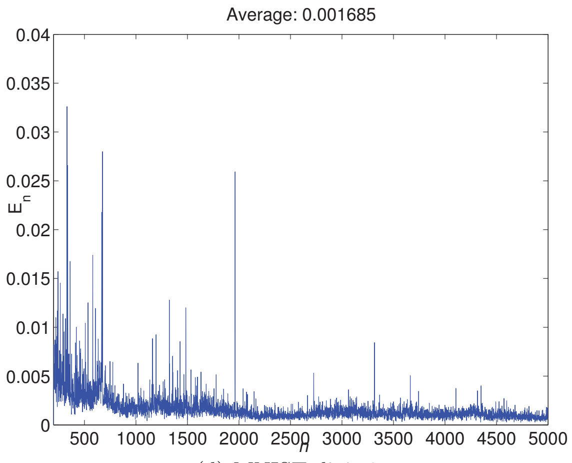
(a) Swiss roll



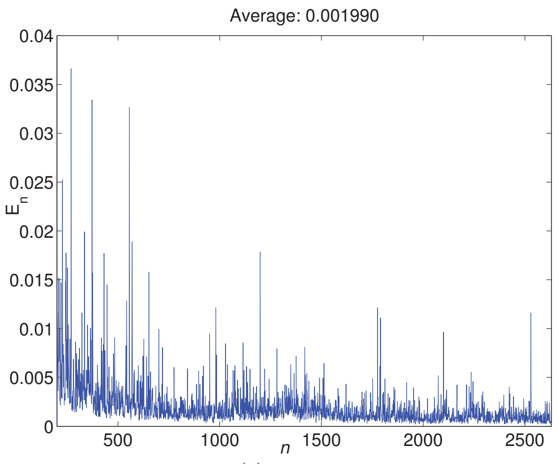
(b) S-curve



(c) Rendered Faces



(d) MNIST digit 2



(e) `ethn`

Figure 3.11: Approximation error ($\mathcal{E}_n$) between the co-ordinates estimated by the incremental landmark ISOMAP and the batch landmark ISOMAP for different numbers of data points (n). It is similar to Figure 3.4, except that incremental landmark ISOMAP is used instead of the basic ISOMAP.

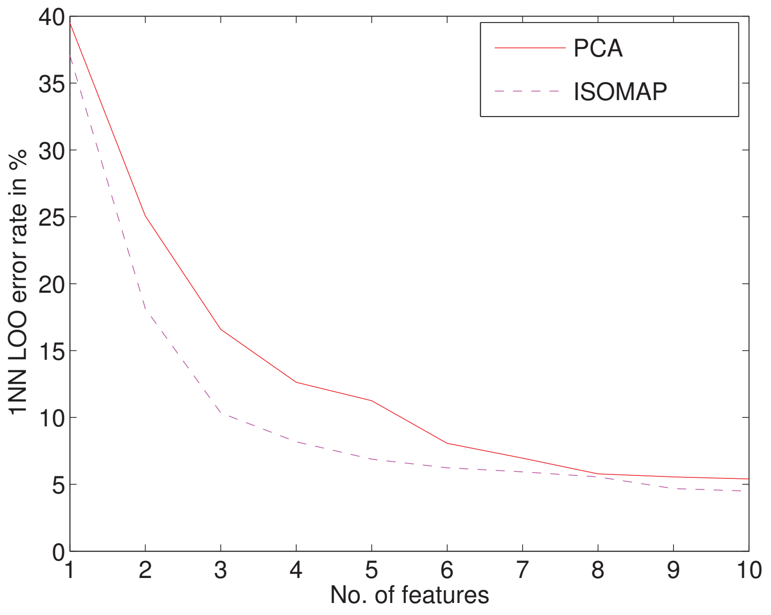


Figure 3.12: Classification performance on `ethn` database, landmark ISOMAP.

the number of possible invariance transformations is infinite. This unboundedness calls for an incremental algorithm, which can accumulate the effect of the data generated. This idea has been exploited in [235] for improving the accuracy in digit classification. Given a digit image, simple distortions like translation, rotation, and skewing are applied to create additional training data for improving the invariance property of a neural network.

We tested a similar idea using the proposed incremental ISOMAP. The training data were generated by first randomly selecting an image from 500 digit "2" images in the MNIST training set. The image was then rotated randomly by θ degree, where θ was uniformly distributed in $[-30, 30]$. The image was used as the input for the incremental landmark ISOMAP with 40 landmarks and a memory size of 10000, with vertex contraction enabled. The training was stopped when 60000 training images were generated. We wanted to investigate how well the rotation angle is recovered by the nonlinear mapping. This was done by using an independent set of digit "2" images from the MNIST testing set, which was of size 1032. For each image $\tilde{\mathbf{y}}^{(i)}$, it was rotated by 15 different angles: $30j/7$ for $j = -7, \ldots, 7$. The mappings of these 15 images, $\tilde{\mathbf{x}}_{-7}^{(i)}, \ldots, \tilde{\mathbf{x}}_{7}^{(i)}$, were found using the out-of-sample extension of ISOMAP. If ISOMAP can discover the rotation angle, there should exist a linear projection direction $\mathbf{h}$ such that $\mathbf{h}^T \tilde{\mathbf{x}}_l^{(i)} \approx c_i + l$ for all i and l, where c_i is a constant specific to $\tilde{\mathbf{y}}^{(i)}$. This is equivalent to

$$\mathbf{h}^T \left(\tilde{\mathbf{x}}_l^{(i)} - \tilde{\mathbf{x}}_0^{(i)} \right) \approx l, \tag{3.8}$$

which is an over-determined linear system. The goodness of the mapping $\tilde{\mathbf{x}}_l^{(i)}$ in terms of how well the rotation angle is recovered can thus be quantified by the residue of the above equation. For comparison, a similar procedure was applied for PCA using the first 10000 generated images. Figure 3.14 shows the result. We can see that the residue for ISOMAP is smaller than PCA, indicating that ISOMAP recovers the rotation angle better. The residue is even smaller when additional images are generated to improve the mapping.

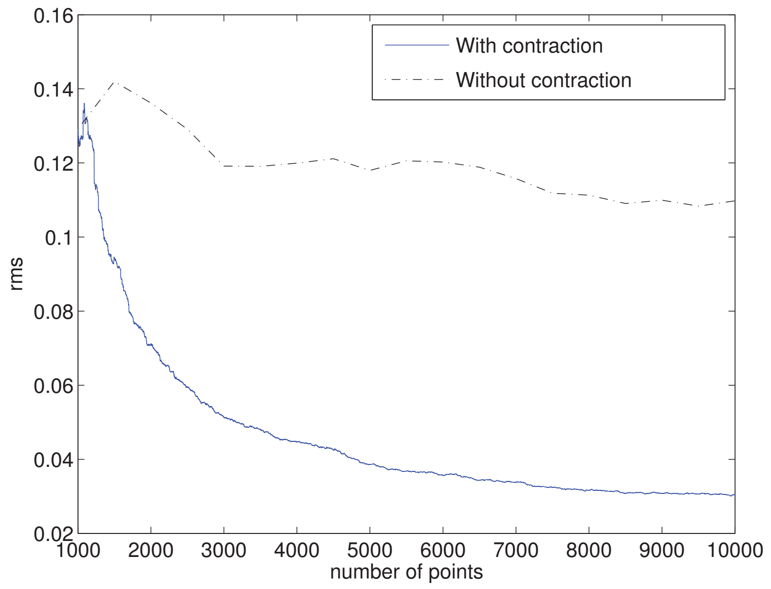


Figure 3.13: Utility of vertex contraction. Solid line: the root-mean-square error (when compared with the ground truth) of the geodesic distance estimate for points currently held in memory when vertex contraction is used. Dash-dot line: the corresponding root-mean-square error when the new points are stored in the memory instead of being contracted.

3.4 Discussion

We have presented algorithms to incrementally update the co-ordinates produced by ISOMAP. Our approach can be extended to other manifold learning algorithms; for example, creating an incremental version of Laplacian eigenmap requires the update of the neighborhood graph and the leading eigenvectors of a matrix (graph Laplacian) derived from the neighborhood graph.

The convergence of geodesic distance is guaranteed since the geodesic distances are maintained exactly. Subspace iteration used in co-ordinate update is provably convergent if a sufficient number of iterations is used, assuming all eigenvalues are simple, which is generally the case. The fact that we only run subspace iteration once can be interpreted as trading off guaranteed convergence with empirical efficiency. Since the change in target inner product matrix is often small, the eigenvector improvement due to subspace iterations with different number of points is aggregated, leading to the low approximation error as shown in Figures 3.4 and 3.11.

While running the proposed incremental ISOMAP is much faster than running the batch version repeatedly, it is more efficient to run the batch version once using all the data points if only the final solution is desired (compare Tables 3.1 and 3.2, as well as Tables 3.3 and 3.4). It is because maintaining intermediate geodesic distances and co-ordinates accurately requires extra computation. The incremental algorithm can be made faster if the geodesic distances are updated upon seeing p subsequent points, $p > 1$. We first embed $\mathbf{y}_{n+1}, \ldots, \mathbf{y}_{n+p}$ independently by the method in section 3.2.1.3. The geodesic distances among the existing points are not updated, and the same set of $\mathbf{x}_i$ is used to find $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+p}$. After that, all the geodesic distances are updated, followed by the update of $\mathbf{x}_1, \ldots, \mathbf{x}_{n+p}$ by subspace iteration. This strategy makes the incremental algorithm almost p-times faster, because the time to embed the new points is very small (see the time for "computing $\mathbf{x}_{n+1}$" in Tables 3.1 and 3.3). On the other hand, the quality of the embedding will deteriorate because the embedding of the existing points cannot benefit from the new points. This

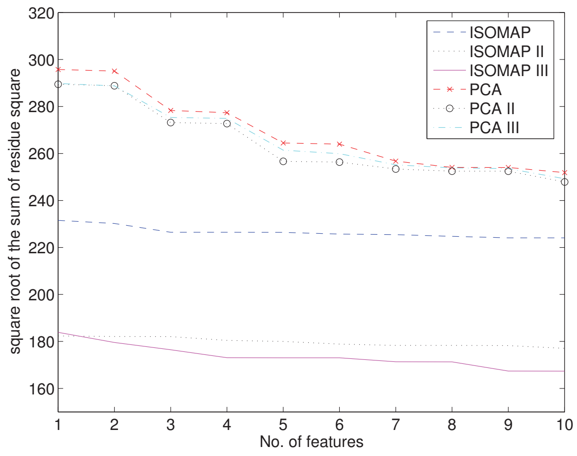


Figure 3.14: Sum of residue square for 1032 images in 15 rotation angles. The larger the residue, the worse the representation. “PCA” and “ISOMAP” correspond to the nonlinear mapping obtained by PCA and ISOMAP when 10000 generated images are used for training, respectively. “ISOMAP II”/“PCA II” and “ISOMAP III”/“PCA III” correspond to the result when the learning stops after 20000 and 50000 images are generated, respectively.

strategy is particularly attractive with large n, because the effect of $\mathbf{y}_{n+1}, \ldots, \mathbf{y}_{n+p}$ on $\mathbf{y}_{n+p+1}$ is small.

Also, for a fixed amount of memory, the solution obtained by the incremental version can be superior to that of the batch version. This is because the incremental version can perform vertex contraction, thereby obtaining a better geodesic distance estimate. The incremental version can be easily adopted to an unbounded data stream when training data are generated by applying invariance transformation, too.

3.4.1 Variants of the Main Algorithms

Our incremental algorithm can be modified to cope with variable neighborhood definition, if the user is willing to do some tedious book-keeping. We can, for example, use ϵ-neighborhood with the value of ϵ re-adjusted whenever, say, 200 data points have arrived. This can be easily achieved by first calculating the edges that need to be deleted or added because of the new neighborhood definition. The algorithms in sections 3.2.1 and 3.2.2 are then used to update the geodesic distances. The embedded co-ordinates can then be updated accordingly.

The supervised ISOMAP algorithm in [276], which utilizes a criterion similar to the Fisher discriminant for embedding, can also be converted to become incremental. The only change is that the subspace iteration method for solving a generalized eigenvalue problem is used instead. The proposed incremental ISOMAP can be easily converted to incremental conformal ISOMAP [55]. In conformal ISOMAP, the edge weight w_{ij} is $\Delta_{ij}/\sqrt{M(i)M(j)}$, where $M(i)$ denotes the distance of $\mathbf{y}_i$ from its k nearest neighbors. The computation of the shortest path distances and eigen-decomposition remains the same. To convert this to its incremental counterpart, we need to maintain the sum of the weights of the k nearest neighbors of different vertices. The change in the edge weights due to the insertion and deletion of edges as a new point comes can be easily tracked. The target inner product matrix is updated, and subspace iteration can be used to update the embedding.

3.4.2 Comparison With Out-of-sample Extension

One problem closely related to incremental nonlinear mapping is the "out-of-sample extension" [17]: given the embedding $\mathbf{x}_1, \ldots, \mathbf{x}_n$ for a "training set" $\mathbf{y}_1, \ldots, \mathbf{y}_n$, what is the embedding result $(\mathbf{x}_{n+1})$ for a "testing" point $\mathbf{y}_{n+1}$? This is effectively the problem considered in section 3.2.1.3. In incremental learning, however, we go beyond obtaining $\mathbf{x}_{n+1}$: the co-ordinate estimates $\mathbf{x}_1, \ldots, \mathbf{x}_n$ of the existing points are also improved by $\mathbf{y}_{n+1}$. In the case of incremental ISOMAP, this amounts to updating the geodesic distances and then applying subspace iteration.

The out-of-sample extension is faster because it skips the improvement step. However, it is less accurate, and cannot provide intermediate embedding with good quality as points are accumulated. Incremental ISOMAP, on the other hand, utilizes the new samples to continuously improve the co-ordinate estimates. Out-of-sample extension may be more appealing when a large number of samples have been accumulated and the geodesic distances and $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are reasonably accurate. Even in this case, though, the strategy of updating $\mathbf{x}_1, \ldots, \mathbf{x}_n$ after p new points (with $p > 1$) have been embedded works equally as well. The updating of geodesic distances and co-ordinates occurs infrequently in this case, and its amortized computational cost is very low.

Incremental ISOMAP is also preferable to out-of-sample extension when there is a drifting of data characteristics. In out-of-sample extension, the n points collected are assumed to be representative of all future data points that are likely to be observed. There is no way to capture the change of data characteristics. In incremental ISOMAP, however, we can easily maintain an embedding using a window of the points recently encountered. Changes in data characteristics are captured as the geodesic distances and co-ordinate estimates are updated. Vertex contraction should be turned off if incremental ISOMAP is run in this mode, to ensure that the effect of old data points is erased.

3.4.3 Implementation Details

The subspace iteration in section 3.2.1.4 requires that the eigenvalues corresponding to the leading eigenvectors have the largest absolute values. This can be violated if the target inner product matrix has a large negative eigenvalue. To tackle this, we shift the spectrum and find the eigenvectors of $(\mathbf{B}+\alpha\mathbf{I})$ instead of $\mathbf{B}$. Subspace iteration on $(\mathbf{B}+\alpha\mathbf{I})$ can proceed in almost the same manner, because $(\mathbf{B}+\alpha\mathbf{I})\mathbf{v} = \mathbf{B}+\alpha\mathbf{v}$. While a large value of α guarantees that all shifted eigenvalues are positive, this has the adverse effect of reducing the rate of convergence of the eigenvectors, because the shift reduces the ratio between adjacent eigenvalues. We empirically set $\alpha = \max(-0.7\lambda_{\min}(\mathbf{B}) - 0.3\lambda_{d\text{-th}}(\mathbf{B}), 0)$, where $\lambda_{\min}(\mathbf{B})$ and $\lambda_{d\text{-th}}(\mathbf{B})$ denote the smallest (most negative) and the d-th largest eigenvalues, respectively. The later is being maintained by the incremental algorithm, while the former can be found by, say, residual norm bounds or Gerschgoren disk bounds. In practice, $\lambda_{\min}(\mathbf{B})$ is found at the initialization stage. This estimate is updated only when a large number of data points have been accumulated.

During the incremental learning, the neighborhood graph may be temporarily disconnected. A simple solution is to embed only the largest graph component. The excluded vertices are added back for embedding again when they become reconnected as additional data points are encountered. Alternatively, an edge can be added between the two nearest vertices to connect the two disconnected components in the neighborhood graph.

3.5 Summary

Nonlinear dimensionality reduction is an important problem with applications in pattern recognition, computer vision, and machine learning. We have developed an algorithm for the incremental nonlinear mapping problem by modifying the well-known ISOMAP algorithm. The core idea is to efficiently update the geodesic distances (a graph theoretic problem) and re-estimate the eigenvectors (a numerical analysis problem), using the previous computation results. Our experiments on synthetic data as well as real world images validate that the proposed method is almost as accurate as running the batch version, while saving significant computation time. Our algorithm can also be easily adopted to other manifold learning methods to produce their incremental versions.

Chapter 4

Simultaneous Feature Selection and Clustering

Hundreds of clustering algorithms have been proposed in the literature for clustering in different applications. In this chapter, we examine a different aspect of clustering that is often neglected: the issue of feature selection. Our focus will be on partitional clustering by a mixture of Gaussians, though the method presented here can be easily generalized to other types of mixtures. We are interested in mixture-based clustering because its statistical nature gives us a solid foundation for analyzing its behavior. Also, it leads to good results in many cases. We propose the concept of *feature saliency* and introduce an *expectation-maximization* (EM) algorithm to estimate it, in the context of mixture-based clustering. We adopt the *minimum message length* (MML) model selection criterion, so the saliency of irrelevant features is driven towards zero, which corresponds to performing feature selection. The MML criterion and the EM algorithm are then extended to simultaneously estimate the feature saliencies and the number of clusters.

The remainder of this chapter is organized as follows. We discuss the challenge of feature selection in unsupervised domain in Section 4.1. In Section 4.2, we review previous attempts to solve the feature selection problem in unsupervised learning. The details of our approach are presented in Section 4.3. Experimental results are reported in Section 4.4, followed by comments on the proposed algorithm in Section 4.5. Finally, we conclude in Section 4.6.

4.1 Clustering and Feature Selection

Clustering, similar to supervised classification and regression, can be benefited by using a good subset of the available features. One simple example illustrating the corrupting influence of irrelevant features can be seen in Figure 4.1, where the irrelevant feature makes it hard for the algorithm in [81] to discover the two underlying clusters. Feature selection has been widely studied in the context of supervised learning (see [101, 26, 122, 151, 153] and references therein, and also section 1.2.3.1), where the ultimate goal is to select features that can achieve the highest accuracy on unseen data. Feature selection has received comparatively very little attention in unsupervised learning or clustering. One important reason is that it is not at all clear how to assess the relevance of a subset of features without resorting to class labels. The problem is made even more challenging when the number of clusters is unknown, since the optimal number of clusters and the optimal feature subset are inter-related, as illustrated in Figure 4.2 (taken from [69]). Note that methods based on variance (such as *principal components analysis*) need not select good features for clustering, as features with large variance can

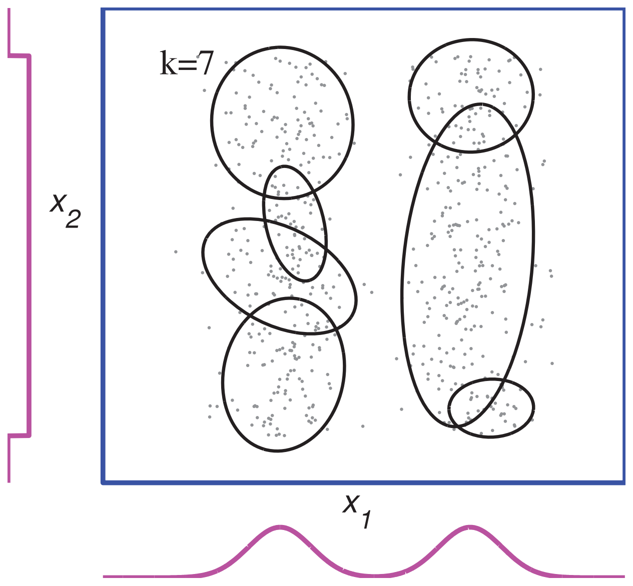


Figure 4.1: An irrelevant feature (x_2) makes it difficult for the Gaussian mixture learning algorithm in [81] to recover the two underlying clusters. Gaussian mixture fitting finds seven clusters when both the features are used, but identifies only two clusters when the feature x_1 is used. The curves along the horizontal and vertical axes of the figure indicate the marginal distribution of x_1 and x_2, respectively.

be independent of the intrinsic grouping of the data (see Figure 4.3). Another important problem in clustering is the determination of the number of clusters, which clearly impacts and is influenced by the feature selection issue. Most feature selection algorithms (such as [36, 151, 209]) involve a combinatorial search through the space of all feature subsets. Usually, heuristic (non-exhaustive) methods have to be adopted, because the size of this space is exponential in the number of features. In this case, one generally loses any guarantee of optimality of the selected feature subset.

We propose a solution to the feature selection problem in unsupervised learning by casting it as an estimation problem, thus avoiding any combinatorial search. Instead of selecting a subset of features, we estimate a set of real-valued (actually in [0, 1]) quantities (one for each feature), which we call the *feature saliencies*. This estimation is carried out by an EM algorithm derived for the task. Since we are in the presence of a model-selection-type problem, it is necessary to avoid the situation where all the features are completely salient. This is achieved by adopting a *minimum message length* (MML, [264, 265]) penalty, as was done in [81] to select the number of clusters. The MML criterion encourages the saliencies of the irrelevant features to go to zero, allowing us to prune the feature set. Finally, we integrate the process of feature saliency estimation into the mixture fitting algorithm proposed in [81], thus obtaining a method that is able to simultaneously perform feature selection and determine the number of clusters.

This chapter is based on our journal publication in [163].

4.2 Related Work

Most of the literature on feature selection pertains to supervised learning (see Section 1.2.3.1). Comparatively, not much work has been done for feature selection in unsupervised learning. Of course, any method conceived for supervised learning that does not use the class labels could be

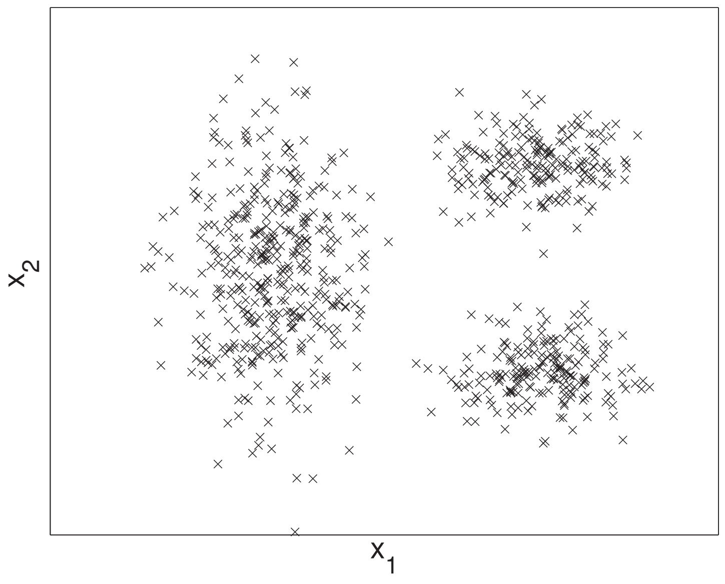


Figure 4.2: The number of clusters is inter-related with the feature subset used. The optimal feature subsets for identifying 3, 2, and 1 clusters in this data set are $\{x_1, x_2\}$, $\{x_1\}$, and $\{x_2\}$, respectively. On the other hand, the optimal number of clusters for feature subsets $\{x_1, x_2\}$, $\{x_1\}$, and $\{x_2\}$ are also 3, 2, and 1, respectively.

used for unsupervised learning; this is the case for methods that measure feature similarity to detect redundant features, using, *e.g.*, mutual information [221] or a maximum information compression index [188]. In [70, 71], the normalized log-likelihood and cluster separability are used to evaluate the quality of clusters obtained with different feature subsets. Different feature subsets and different numbers of clusters, for multinomial model-based clustering, are evaluated using marginal likelihood and cross-validated likelihood in [254]. The algorithm described in [218] uses a LASSO-based idea to select the appropriate features. In [51], the clustering tendency of each feature is assessed by an entropy index. A genetic algorithm is used in [146] for feature selection in k-means clustering. In [246], feature selection for symbolic data is addressed by assuming that irrelevant features are uncorrelated with the relevant features. Reference [60] describes the notion of "category utility" for feature selection in a conceptual clustering task. The CLIQUE algorithm [2] is popular in the data mining community, and it finds hyper-rectangular shaped clusters using a subset of attributes for a large database. The wrapper approach can also be adopted to select features for clustering; this has been explored in our earlier work [82, 165].

All the methods referred to above perform "hard" feature selection (a feature is either selected or not). There are also algorithms that assign weights to different features to indicate their significance. In [190], weights are assigned to different groups of features for k-means clustering based on a score related to the Fisher discriminant. Feature weighting for k-means clustering is also considered in [187], but the goal there is to find the best description of the clusters, after they are identified. The method described in [204] can be classified as learning feature weights for conditional Gaussian networks. An EM algorithm based on Bayesian shrinking is proposed in [100] for unsupervised learning.

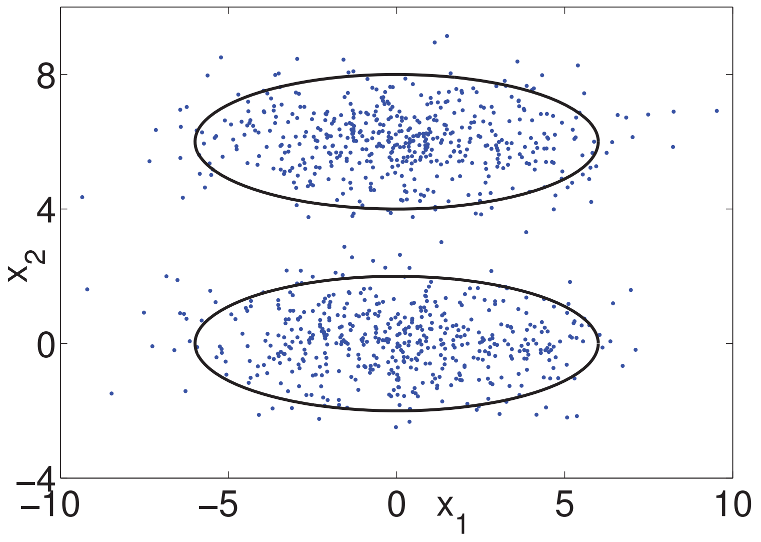


Figure 4.3: Deficiency of variance-based method for feature selection. Feature x_1, although it explains more data variance than feature x_2, is spurious for the identification of the two clusters in this data set.

4.3 EM Algorithm for Feature Saliency

In this section, we propose an EM algorithm for performing mixture-based (or model-based) clustering with feature selection. In mixture-based clustering, each data point is modelled as having been generated by one of a set of probabilistic models [125, 183]. Clustering is then done by learning the parameters of these models and the associated probabilities. Each pattern is assigned to the mixture component that most likely generated it. Although the derivations below refer to Gaussian mixtures, they can be generalized to other types of mixtures.

4.3.1 Mixture Densities

A finite mixture density with k components is defined by

$$p(\mathbf{y}) = \sum_{j=1}^{k} \alpha_j \, p(\mathbf{y}|\boldsymbol{\theta}_j), \tag{4.1}$$

where $\forall_j, \alpha_j \geq 0$; $\sum_j \alpha_j = 1$; each $\boldsymbol{\theta}_j$ is the set of parameters of the j-th component (all components are assumed to have the same form, *e.g.*, Gaussian); and $\boldsymbol{\theta} \equiv \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_k, \alpha_1, ..., \alpha_k\}$ will denote the full parameter set. The goal of mixture estimation is to infer $\boldsymbol{\theta}$ from a set of n data points $\mathcal{Y} = \{\mathbf{y}_1, ..., \mathbf{y}_n\}$, assumed to be samples of a distribution with density given by (4.1). Each $\mathbf{y}_i$ is a d-dimensional feature vector $[y_{i1}, ..., y_{id}]^T$. In the sequel, we will use the indices i, j and l to run through data points (1 to n), mixture components (1 to k), and features (1 to d), respectively.

As is well known, neither the *maximum likelihood* (ML) estimate,

$$\widehat{\boldsymbol{\theta}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \{\log p(\mathcal{Y}|\boldsymbol{\theta})\},$$

nor the *maximum a posteriori* (MAP) estimate (given some prior $p(\boldsymbol{\theta})$)

$$\widehat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \{\log p(\mathcal{Y}|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\},$$

can be found analytically. The usual choice is the EM algorithm, which finds local maxima of these criteria [183]. This algorithm is based on a set $\mathcal{Z} = \{\mathbf{z}_1, ..., \mathbf{z}_n\}$ of n *missing* (latent) labels, where $\mathbf{z}_i = [z_{i1}, ..., z_{ik}]$, with $z_{ij} = 1$ and $z_{ip} = 0$, for $p \neq j$, meaning that $\mathbf{y}_i$ is a sample of $p(\cdot|\boldsymbol{\theta}_j)$. For brevity of notation, sometimes we write $z_i = j$ for such $\mathbf{z}_i$. The complete data log-likelihood, *i.e.*, the log-likelihood if $\mathcal{Z}$ were observed, is

$$\log p(\mathcal{Y}, \mathcal{Z}|\boldsymbol{\theta}) = \sum_{i=1}^{n}\sum_{j=1}^{k} z_{ij} \log\left[\alpha_j p(\mathbf{y}_i|\boldsymbol{\theta}_j)\right]. \tag{4.2}$$

The EM algorithm produces a sequence of estimates $\{\widehat{\boldsymbol{\theta}}(t),\ t = 0, 1, 2, ...\}$ using two alternating steps:

• **E-step:** Compute $\mathcal{W} = E[\mathcal{Z}|\mathcal{Y}, \widehat{\boldsymbol{\theta}}(t)]$, the expected value of the missing data given the current parameter estimate, and plug it into log $p(\mathcal{Y}, \mathcal{Z}|\boldsymbol{\theta})$, yielding the so-called Q-function $Q(\boldsymbol{\theta}, \widehat{\boldsymbol{\theta}}(t)) = \log p\,(\mathcal{Y}, \mathcal{W}|\,\boldsymbol{\theta})$. Since the elements of $\mathcal{Z}$ are binary, we have

$$w_{i,j} \equiv E\left[z_{ij}|\,\mathcal{Y}, \widehat{\boldsymbol{\theta}}(t)\right] = \Pr\left[z_{ij} = 1|\,\mathbf{y}_i, \widehat{\boldsymbol{\theta}}(t)\right] = \frac{\widehat{\alpha}_j(t)\, p(\mathbf{y}_i|\widehat{\boldsymbol{\theta}}_j(t))}{\sum_{j=1}^{k} \widehat{\alpha}_j(t)\, p(\mathbf{y}_i|\widehat{\boldsymbol{\theta}}_k(t))}. \tag{4.3}$$

Notice that α_j is the *a priori* probability that $z_{ij} = 1$ (*i.e.*, that $\mathbf{y}_i$ belongs to cluster j), while w_{ij} is the corresponding *a posteriori* probability, after observing $\mathbf{y}_i$.

• **M-step:** Update the parameter estimates,

$$\widehat{\boldsymbol{\theta}}(t+1) = \arg\max_{\boldsymbol{\theta}} \{Q(\boldsymbol{\theta}, \widehat{\boldsymbol{\theta}}(t)) + \log p(\boldsymbol{\theta})\},$$

in the case of MAP estimation, or without $\log p(\boldsymbol{\theta})$ in the ML case.

4.3.2 Feature Saliency

In this section we define the concept of *feature saliency* and derive an EM algorithm to estimate its value. We assume that the features are conditionally independent given the (hidden) component label, that is,

$$p(\mathbf{y}|\boldsymbol{\theta}) = \sum_{j=1}^{k} \alpha_j\, p(\mathbf{y}|\theta_j) = \sum_{j=1}^{k} \alpha_j \prod_{l=1}^{d} p(y_l|\theta_{jl}), \tag{4.4}$$

where $p(\cdot|\theta_{jl})$ is the pdf of the l-th feature in the j-th component. This assumption enables us to utilize the power of the EM algorithm. In the particular case of Gaussian mixtures, the conditional independence assumption is equivalent to adopting diagonal covariance matrices, which is a common choice for high-dimensional data, such as in naïve Bayes classifiers and latent class models, as well as in the emission densities of continuous hidden Markov models.

Among different definitions of feature irrelevancy (proposed for supervised learning), we adopt the one suggested in [210, 254], which is suitable for unsupervised learning: the l-th feature is irrelevant if its distribution is independent of the class labels, *i.e.*, if it follows a common density,

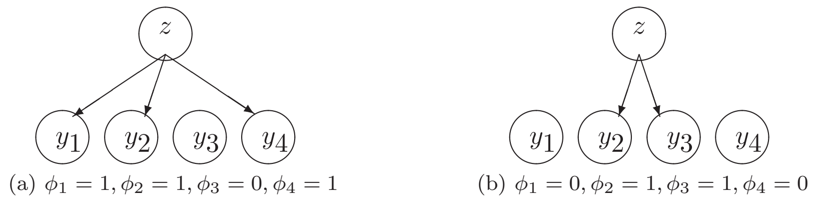

z
y_1
y_2
y_3
y_4
(a) $\phi_1 = 1, \phi_2 = 1, \phi_3 = 0, \phi_4 = 1$
z
y_1
y_2
y_3
y_4
(b) $\phi_1 = 0, \phi_2 = 1, \phi_3 = 1, \phi_4 = 0$

Figure 4.4: An example graphical model for the probability model in Equation (4.5) for the case of four features ($d = 4$) with different indicator variables. $\phi_l = 1$ corresponds to the existence of an arc from z to y_l, and $\phi_l = 0$ corresponds to its absence.

denoted by $q(y_l|\lambda_l)$. Let $\Phi = (\phi_1, ..., \phi_d)$ be a set of d binary parameters, such that $\phi_l = 1$ if feature l is relevant and $\phi_l = 0$, otherwise. The mixture density in (4.4) can then be re-written as

$$p(\mathbf{y}|\Phi, \{\alpha_j\}, \{\theta_{jl}\}, \{\lambda_l\}) = \sum_{j=1}^{k} \alpha_j \prod_{l=1}^{d} [p(y_l|\theta_{jl})]^{\phi_l} [q(y_l|\lambda_l)]^{1-\phi_l}. \tag{4.5}$$

A related model for feature selection in supervised learning has been considered in [197, 210]. Intuitively, Φ determines which edges exist between the hidden label z and the individual features y_l in the graphical model illustrated in Figure 4.4, for the case $d = 4$.

Our notion of *feature saliency* is summarized in the following steps: (i) we treat the ϕ_l's as missing variables; (ii) we define the *feature saliency* as $\rho_l = p(\phi_l = 1)$, the probability that the l-th feature is relevant. This definition makes sense, as it is difficult to know for sure that a certain feature is irrelevant in unsupervised learning. The resulting model (likelihood function) is written as

$$p(\mathbf{y}|\boldsymbol{\theta}) = \sum_{j=1}^{k} \alpha_j \prod_{l=1}^{d} \big(\rho_l p(y_l|\theta_{jl}) + (1 - \rho_l) q(y_l|\lambda_l)\big), \tag{4.6}$$

where $\boldsymbol{\theta} = \{\{\alpha_j\}, \{\theta_{jl}\}, \{\lambda_l\}, \{\rho_l\}\}$ is the set of all the parameters of the model.

Equation (4.6) can be derived as follows. We treat $\rho_l = p(\phi_l = 1)$ as a set of parameters to be estimated (the feature saliencies). We assume the ϕ_l's are mutually independent and also independent of the hidden component label z for any pattern $\mathbf{y}$. Thus,

$$\begin{aligned} p(\mathbf{y}, \Phi) &= p(\mathbf{y}|\Phi)p(\Phi) \\ &= \Big(\sum_{j=1}^{k} \alpha_j \prod_{l=1}^{d} (p(y_l|\theta_{j\,l}))^{\phi_l} (q(y_l|\lambda_l))^{1-\phi_l}\Big) \prod_{l=1}^{d} \rho_l^{\phi_l} (1 - \rho_l)^{1-\phi_l} \\ &= \sum_{j=1}^{k} \alpha_j \prod_{l=1}^{d} (\rho_l p(y_l|\theta_{j\,l}))^{\phi_l} ((1 - \rho_l) q(y_l|\lambda_l))^{1-\phi_l}. \end{aligned} \tag{4.7}$$

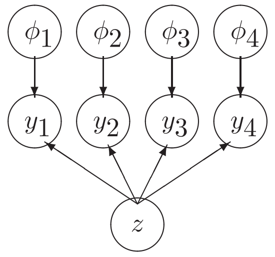


Figure 4.5: An example graphical model showing the mixture density in Equation (4.6). The variables z, $\phi_1, \phi_2, \phi_3, \phi_4$ are "hidden" and only y_1, y_2, y_3, y_4 are observed.

The marginal density for $\mathbf{y}$ is

$$
\begin{aligned}
p(\mathbf{y}) = \sum_{\Phi} p(\mathbf{y}, \Phi) &= \sum_{j=1}^{d} \alpha_j \sum_{\Phi} \prod_{l=1}^{d} (\rho_l p(y_l|\theta_{j\,l}))^{\phi_l} ((1-\rho_l) q(y_l|\lambda_l))^{1-\phi_l} \\
&= \sum_{j=1}^{k} \alpha_j \prod_{l=1}^{d} \sum_{\phi_l=0}^{1} (\rho_l p(y_l|\theta_{j\,l}))^{\phi_l} ((1-\rho_l) q(y_l|\lambda_l))^{1-\phi_l} \qquad (4.8) \\
&= \sum_{j=1}^{k} \alpha_j \prod_{l=1}^{d} \big(p(y_l|\theta_{j\,l})\rho_l + q(y_l|\lambda_l)(1-\rho_l)\big),
\end{aligned}
$$

which is just Equation (4.6). Another way to see how Equation (4.6) is obtained is to notice that the conditional density of y_l given $z = j$ and ϕ_l, $[p(y_l|\theta_{jl})]^{\phi_l} [q(y_l|\lambda_l)]^{1-\phi_l}$, can be written as $\phi_l\, p(y_l|\theta_{jl}) + (1-\phi_l)\, q(y_l|\lambda_l)$, because ϕ_l is binary. Taking the expectation with respect to ϕ_l and z leads to Equation (4.6).

The form of $q(.|.)$ reflects our prior knowledge about the distribution of the non-salient features. In principle, it can be any 1-D distribution (*e.g.*, a Gaussian, a student-t, or even a mixture). We shall limit $q(.|.)$ to be a Gaussian, since this leads to reasonable results in practice.

Equation (4.6) has a generative interpretation. As in a standard finite mixture, we first select the component label j by sampling from a multinomial distribution with parameters $(\alpha_1, \ldots, \alpha_k)$. Then, for each feature $l = 1, ..., d$, we flip a biased coin whose probability of getting a head is ρ_l; if we get a head, we use the mixture component $p(.|\theta_{jl})$ to generate the l-th feature; otherwise, the common component $q(.|\lambda_l)$ is used. A graphical model representation of Equation (4.6) is shown in Figure 4.5 for the case $d = 4$.

4.3.2.1 EM Algorithm

By treating $\mathcal{Z}$ (the hidden class labels) and Φ (the feature indicators) as hidden variables, one can derive an EM algorithm for parameter estimation. The complete-data log-likelihood for the model in Equation (4.6) is

$$
P(\mathbf{y}_i, z_i = j, \Phi) = \alpha_j \prod_{l=1}^{d} (\rho_l p(y_{i\,l}|\theta_{j\,l}))^{\phi_l} ((1-\rho_l) q(y_{i\,l}|\lambda_l))^{1-\phi_l}. \qquad (4.9)
$$

Define the following quantities:

$$w_{ij} = p(z_i = j|\mathbf{y}_i), \qquad u_{ij\,l} = p(z_i = j, \phi_l = 1|\mathbf{y}_i), \qquad v_{ij\,l} = p(z_i = j, \phi_l = 0|\mathbf{y}_i).$$

They are calculated using the current parameter estimate $\boldsymbol{\theta}^{\text{now}}$. Note that $(u_{ij\,l} + v_{ij\,l}) = w_{ij}$ and $\sum_{i=1}^{n}\sum_{j=1}^{k} w_{ij} = n$. The expected complete data log-likelihood based on $\boldsymbol{\theta}^{\text{now}}$ is

$$\begin{aligned}
&E_{\boldsymbol{\theta}^{\text{now}}}[\log p(\mathcal{Y}, \mathbf{z}, \Phi)] \\
&= \sum_{i,j,\Phi} p(z_i = j, \Phi|\mathbf{y}_i)\Big(\log\alpha_j + \sum_l \big(\phi_l(\log p(y_{il}|\theta_{j\,l}) + \log\rho_l) \\
&\qquad + (1-\phi_l)(\log q(y_{il}|\lambda_l) + \log(1-\rho_l))\big)\Big) \\
&= \sum_{i,j} p(z_i = j|\mathbf{y}_i)\log\alpha_j + \sum_{i,j}\sum_l \sum_{\phi_l=0}^{1} p(z_i = j, \phi_l|\mathbf{y}_i)\big(\phi_l(\log p(y_{il}|\theta_{j\,l}) + \log\rho_l) \\
&\qquad + (1-\phi_l)(\log q(y_{il}|\lambda_l) + \log(1-\rho_l))\big) \\
&= \underbrace{\sum_j(\sum_i w_{ij})\log\alpha_j}_{\text{part 1}} + \sum_{j,l}\underbrace{\sum_i u_{ij\,l}\log p(y_{il}|\theta_{j\,l})}_{\text{part 2}} + \sum_l\underbrace{\sum_{i,j} v_{ij\,l}\log q(y_{il}|\lambda_l)}_{\text{part 3}} \\
&\qquad + \sum_l \underbrace{\Big(\log\rho_l\sum_{i,j} u_{ij\,l} + \log(1-\rho_l)\sum_{i,j} v_{ij\,l}\Big)}_{\text{part 4}}.
\end{aligned} \tag{4.10}$$

The four parts in the equation above can be maximized separately. Recall that the densities $p(.)$ and $q(.)$ are univariate Gaussian and are characterized by their means and variances. As a result, maximizing the expected complete data log-likelihood leads to the M-step in Equations (4.18)–(4.23). For the E-step, observe that

$$\begin{aligned}
p(\phi_l = 1|z_i = j, \mathbf{y}_i) &= \frac{p(\phi_l = 1, \mathbf{y}_i|z_i = j)}{p(\mathbf{y}_i|z_i = j)} \\
&= \frac{\rho_l p(y_l|\theta_{j\,l})\prod_{l'\neq l}(\rho_{l'}p(y_{l'}|\theta_{j\,l'}) + (1-\rho_{l'})q(y_{l'}|\lambda_{l'}))}{\prod_{l'}(\rho_{l'}p(y_{l'}|\theta_{j\,l'}) + (1-\rho_{l'})q(y_{l'}|\lambda_{l'}))} \\
&= \frac{\rho_l p(y_l|\theta_{j\,l})}{\rho_l p(y_l|\theta_{j\,l}) + (1-\rho_l)q(y_l|\lambda_l)} = \frac{a_{ij\,l}}{c_{ij\,l}}.
\end{aligned}$$

Therefore, equation (4.16) follows because

$$u_{ij,l} = p(\phi_l = 1|z_i = j, \mathbf{y}_i)p(z_i = j|\mathbf{y}_i) = \frac{a_{ij\,l}}{c_{ij\,l}}w_{ij}. \tag{4.11}$$

So, the EM algorithm is

- **E-step:** Compute the following quantities:

$$a_{ij\,l} = p(\phi_l = 1, y_{i\,l}|z_i = j) = \rho_l\, p(y_{i\,l}|\theta_{j\,l}) \tag{4.12}$$

$$b_{ij\,l} = p(\phi_l = 0, y_{i\,l}|z_i = j) = (1 - \rho_l)\, q(y_{i\,l}|\lambda_l) \tag{4.13}$$

$$c_{ij\,l} = p(y_{il}|z_i = j) = a_{ijl} + b_{ijl} \tag{4.14}$$

$$w_{ij} = p(z_i = j|\mathbf{y}_i) = \frac{\alpha_j \prod_l c_{ij\,l}}{\sum_j \alpha_j \prod_l c_{ij\,l}} \tag{4.15}$$

$$u_{ij\,l} = p(\phi_l = 1, z_i = j|\mathbf{y}_i) = \frac{a_{ij\,l}}{c_{ij\,l}} w_{ij} \tag{4.16}$$

$$v_{ij\,l} = p(\phi_l = 0, z_i = j|\mathbf{y}_i) = w_{ij} - u_{ij\,l} \tag{4.17}$$

- **M-step:** Re-estimate the parameters according to following expressions:

$$\widehat{\alpha_j} = \frac{\sum_i w_{ij}}{\sum_{ij} w_{ij}} = \frac{\sum_i w_{ij}}{n} \tag{4.18}$$

$$\widehat{\text{Mean in }\theta}_{j\,l} = \frac{\sum_i u_{ij\,l}\, y_{i\,l}}{\sum_i u_{ijl}} \tag{4.19}$$

$$\widehat{\text{Var in }\theta}_{j\,l} = \frac{\sum_i u_{ij\,l}\, (y_{i\,l} - (\widehat{\text{Mean in }\theta}_{j\,l}))^2}{\sum_i u_{ij\,l}} \tag{4.20}$$

$$\widehat{\text{Mean in }\lambda_l} = \frac{\sum_i (\sum_j v_{ij\,l})\, y_{i\,l}}{\sum_{ij} v_{ij\,l}} \tag{4.21}$$

$$\widehat{\text{Var in }\lambda_l} = \frac{\sum_i (\sum_j v_{ij\,l})(y_{il} - (\widehat{\text{Mean in }\lambda_l}))^2}{\sum_{ij} v_{ij\,l}} \tag{4.22}$$

$$\widehat{\rho_l} = \frac{\sum_{i,j} u_{ij\,l}}{\sum_{i,j} u_{ij\,l} + \sum_{i,j} v_{ij\,l}} = \frac{\sum_{i,j} u_{ij\,l}}{n}. \tag{4.23}$$

In these equations, the variable $u_{ij\,l}$ measures how important the i-th pattern is to the j-th component, when the l-th feature is used. It is thus natural that the estimates of the mean and the variance in θ_{jl} are weighted sums with weight $u_{ij\,l}$. A similar relationship exists between $\sum_j v_{ij\,l}$ and λ_l. The term $\sum_{ij} u_{ij\,l}$ can be interpreted as how likely it is that ϕ_l equals one, explaining why the estimate of ρ_l is proportional to $\sum_{ij} u_{ij\,l}$.

4.3.3 Model Selection

Standard EM for mixtures exhibits some weaknesses, which also affect the EM algorithm introduced above: it requires knowledge of k (the number of mixture components), and a good initialization is essential for reaching a good local optimum (not to mention the global optimum). To overcome these difficulties, we adopt the approach in [81], which is based on the minimum message length (MML) criterion [265, 264].

The MML criterion for our model consists of minimizing, with respect to $\boldsymbol{\theta}$, the following cost function (after discarding the order one term)

$$-\log p(\mathcal{Y}|\boldsymbol{\theta}) + \frac{k+d}{2}\log n + \frac{r}{2}\sum_{l=1}^{d}\sum_{j=1}^{k}\log(n\alpha_j\rho_l) + \frac{s}{2}\sum_{l=1}^{d}\log(n(1-\rho_l)), \tag{4.24}$$

where r and s are the number of parameters in $\theta_{j\,l}$ and λ_l, respectively. If $p(.|.)$ and $q(.|.)$ are univariate Gaussians (arbitrary mean and variance), $r = s = 2$. Equation (4.24) is derived by considering the *minimum message length* (MML) criterion (see [81] for details and references)

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \Big\{ -\log p(\boldsymbol{\theta}) - \log p(\mathcal{Y}|\boldsymbol{\theta}) + \frac{1}{2}\log|\mathbf{I}(\boldsymbol{\theta})| + \frac{c}{2}(1+\log\frac{1}{12}) \Big\}, \tag{4.25}$$

where $\boldsymbol{\theta}$ is the set of parameter of the model, c is the dimension of $\boldsymbol{\theta}$, $\mathbf{I}(\boldsymbol{\theta}) = -E[D^2_{\boldsymbol{\theta}} \log p(\mathcal{Y}|\boldsymbol{\theta})]$ is the (expected) Fisher information matrix (the negative expected value of the Hessian of the log-likelihood), and $|\mathbf{I}(\boldsymbol{\theta})|$ is the determinant of $\mathbf{I}(\boldsymbol{\theta})$. The information matrix for the model (4.6) is very difficult to obtain analytically. Therefore, as in [81], we approximate it by the information matrix of the complete data log-likelihood, $\mathbf{I}_c(\boldsymbol{\theta})$. By differentiating the logarithm of equation (4.9), we can show that

$$\begin{aligned}\mathbf{I}_c(\boldsymbol{\theta}) = \text{block-diag}\Big[&\mathcal{M}, \frac{1}{\rho_1(1-\rho_1)}, \ldots, \frac{1}{\rho_d(1-\rho_d)}, \alpha_1\rho_1\mathbf{I}(\theta_{11}), \ldots, \alpha_1\rho_d\mathbf{I}(\theta_{1d}),\\ &\alpha_2\rho_1\mathbf{I}(\theta_{21}), \ldots, \alpha_2\rho_d\mathbf{I}(\theta_{2d}), \ldots, \alpha_k\rho_1\mathbf{I}(\theta_{21}), \ldots, \alpha_k\rho_d\mathbf{I}(\theta_{2d}),\\ &(1-\rho_1)\mathbf{I}(\lambda_1), \ldots, (1-\rho_d)\mathbf{I}(\lambda_d)\Big],\end{aligned} \tag{4.26}$$

where $\mathcal{M}$ is the information matrix of the multinominal distribution with parameters $(\alpha_1, \ldots, \alpha_k)$. The size of $\mathbf{I}(\boldsymbol{\theta})$ is $(k + d + kdr + ds)$, where r and s are the number of parameters in $\theta_{j\,l}$ and λ_l, respectively. Note that $(\rho_l(1-\rho_l))^{-1}$ is the Fisher information of a Bernoulli distribution with parameter ρ_l. Thus we can write

$$\begin{aligned}\log|\mathbf{I}_c(\boldsymbol{\theta})| = \log\mathbf{I}(\{\alpha_j\}) + \sum_{l=1}^{d}\log\mathbf{I}(\rho_l) + r\sum_{j=1}^{k}\sum_{l=1}^{d}\log(\alpha_j\rho_l)\\ + \sum_{j=1}^{k}\sum_{l=1}^{d}\log\mathbf{I}(\theta_{j\,l}) + s\sum_{l=1}^{d}\log(1-\rho_l) + \sum_{l=1}^{d}\log\mathbf{I}(\lambda_l).\end{aligned} \tag{4.27}$$

For the prior densities of the parameters, we assume that different groups of parameters are independent. Specifically, $\{\alpha_j\}$, ρ_l (for different values of l), $\theta_{j\,l}$ (for different values of j and l) and λ_l (for different values of l) are independent. Furthermore, since we have no knowledge about the parameters, we adopt non-informative Jeffrey's priors (see [81] for details and references), which are proportional to the square root of the determinant of the corresponding information matrices. When we substitute $p(\boldsymbol{\theta})$ and $|\mathbf{I}(\boldsymbol{\theta})|$ into equation (4.25), and drop the order-one term, we obtain our final criterion, which is equation (4.24).

From a parameter estimation viewpoint, Equation (4.24) is equivalent to a *maximum a posteriori* (MAP) estimate,

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \Big\{ \log p(\mathcal{Y}|\boldsymbol{\theta}) - \frac{rd}{2}\sum_{l=1}^{k}\log\alpha_j - \frac{s}{2}\sum_{l=1}^{d}\log(1-\rho_l) - \frac{rk}{2}\sum_{l=1}^{d}\log\rho_l \Big\}, \tag{4.28}$$

with the following (Dirichlet-type, but improper) priors on the α_j's and ρ_l's:

$$\begin{aligned} p(\alpha_1, \ldots, \alpha_k) &\propto \prod_{j=1}^{k} \alpha_j^{-rd/2}, \\ p(\rho_1, \ldots, \rho_d) &\propto \prod_{l=1}^{d} \rho_l^{-rk/2} (1-\rho_l)^{-s/2}. \end{aligned}$$

Since these priors are conjugate with respect to the complete data likelihood, the EM algorithm undergoes a minor modification: the M-step Equations (4.18) and (4.23) are replaced by

$$\widehat{\alpha_j} = \frac{\max(\sum_i w_{ij} - \frac{rd}{2}, 0)}{\sum_j \max(\sum_i w_{ij} - \frac{rd}{2}, 0)} \tag{4.29}$$

$$\widehat{\rho_l} = \frac{\max(\sum_{i,j} u_{ijl} - \frac{kr}{2}, 0)}{\max(\sum_{i,j} u_{ijl} - \frac{kr}{2}, 0) + \max(\sum_{i,j} v_{ijl} - \frac{s}{2}, 0)}. \tag{4.30}$$

In addition to the log-likelihood, the other terms in Equation (4.24) have simple interpretations. The term $\frac{k+d}{2} \log n$ is a standard MDL-type [215] parameter code-length corresponding to k α_j values and d ρ_l values. For the l-th feature in the j-th component, the "effective" number of data points for estimating θ_{jl} is $n\alpha_j\rho_l$. Since there are r parameters in each θ_{jl}, the corresponding code-length is $\frac{r}{2}\log(n\rho_l\alpha_j)$. Similarly, for the l-th feature in the common component, the number of effective data points for estimation is $n(1-\rho_l)$. Thus, there is a term $\frac{s}{2}\log(n(1-\rho_l))$ in (4.24) for each feature.

One key property of Equations (4.29) and (4.30) is their pruning behavior, forcing some of the α_j to go to zero and some of the ρ_l to go to zero or one. This pruning behavior also has the indirect benefit of protecting us from almost singular covariance matrix in a mixture component: the weight of such a component is usually very small, and the component is likely to be pruned in the next few iterations. Concerns that the message length in (4.24) may become invalid at these boundary values can be circumvented by the arguments in [81]: when ρ_l goes to zero, the l-th feature is no longer salient and ρ_l and $\theta_{1l}, \ldots, \theta_{Kl}$ are removed; when ρ_l goes to 1, λ_l and ρ_l are dropped.

Finally, since the model selection algorithm determines the number of components, it can be initialized with a large value of k, thus alleviating the need for a good initialization, as shown in [81]. Because of this, a component-wise version of EM can be adopted [37, 81]. The algorithm is summarized in Algorithm 4.1.

4.3.4 Post-processing of Feature Saliency

The feature saliencies generated by Algorithm 4.1 attempt to find the best way to *model* the data, using different component densities. Alternatively, we can consider feature saliencies that best *discriminate* between different components. This can be more appropriate if the ultimate goal is to discover well-separated clusters. If the components are well-separated, each pattern is likely to be generated by one component only. Therefore, one quantitative measure of the separability of the clusters is

$$J = \sum_{i=1}^{n} \log p(z_i = t_i | \mathbf{y}_i), \tag{4.31}$$

```
**Input:** Training data $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$, minimum number of components $k_{min}$
**Output:** Number of components $k$, mixture parameters $\{\theta_{jl}\}$, $\{\alpha_j\}$, parameters of common distri-
  bution $\{\lambda_l\}$ and feature saliencies $\{\rho_l\}$
  {Initialization}
  Set the parameters of a large number of mixture components randomly
  Set the common distribution to cover all data
  Set the feature saliency of all features to 0.5
  {Initialization ends; main loop begins}
  **while** $k > k_{min}$ **do**
    **while** not reach local minimum **do**
      Perform E-step according to Equations (4.12) to (4.17)
      Perform M-step according to Equations (4.19) to (4.22), (4.29) and (4.30)
      If $\alpha_j$ becomes zero, the $j$-th component is pruned.
      If $\rho_l$ becomes 1, $q(y_l|\lambda_l)$ is pruned.
      If $\rho_l$ becomes 0, $p(y_l|\theta_{jl})$ are pruned for all $j$
    **end while**
    Record the current model parameters and its message length
    Remove the component with the smallest weight
  **end while**
  Return the model parameters that yield the smallest message length
```

Algorithm 4.1: The unsupervised feature saliency algorithm.

where $t_i = \arg\max_j p(z_i = j|\mathbf{y}_i)$. Intuitively, J is the sum of the logarithms of the posterior probabilities of the data, assuming that each data point was indeed generated by the component with maximum posterior probability (an implicit assumption in mixture-based clustering). J can then be maximized by varying ρ_l while keeping the other parameters fixed.

Unlike the MML criterion, J cannot be optimized by an EM algorithm. However, by defining

$$h_{i\,lj} = \frac{p(y_{il}|\theta_{j\,l}) - q(y_l|\lambda_l)}{\rho_l p(y_{il}|\theta_{j\,l}) + (1-\rho_l) q(y_{il}|\lambda_l)},$$

$$g_{i\,l} = \sum_{j=1}^{k} w_{ij} h_{i\,lj},$$

we can show that

$$\frac{\partial}{\partial \rho_l} \log w_{ij} = h_{i\,lj} - g_{i\,l},$$

$$\frac{\partial^2}{\partial \rho_l \partial \rho_m} \log w_{ij} = \sum_{i=1}^{n} \Big(g_{i\,l} g_{im} - \sum_{j=1}^{k} w_{ij} h_{i\,lj} h_{imj} \Big), \qquad \text{for } l \neq m,$$

$$\frac{\partial^2}{\partial \rho_l^2} \log w_{ij} = \sum_{i=1}^{n} (g_{i\,l}^2 - h_{i\,lj}^2).$$

The gradient and Hessian of J can then be calculated accordingly, if we ignore the dependence of t_i on ρ_l. We can then use any constrained non-linear optimization software to find the optimal values of ρ_l in $[0, 1]$. We have used the MATLAB optimization toolbox in our experiments. After obtaining the set of optimized ρ_l, we fix them and estimate the remaining parameters using the EM algorithm.

4.4 Experimental Results

4.4.1 Synthetic Data

The first synthetic data set consisted of 800 data points from a mixture of four equiprobable Gaussians $\mathcal{N}(\mathbf{m}_i, \mathbf{I}), i = \{1, 2, 3, 4\}$, where $\mathbf{m}_1 = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$, $\mathbf{m}_2 = \begin{bmatrix} 1 \\ 9 \end{bmatrix}$, $\mathbf{m}_3 = \begin{bmatrix} 6 \\ 4 \end{bmatrix}$, $\mathbf{m}_4 = \begin{bmatrix} 7 \\ 10 \end{bmatrix}$ (Figure 4.6(a)). Eight "noisy" features (sampled from a $\mathcal{N}(0, 1)$ density) were then appended to this data, yielding a set of 800 10-dimensional patterns. We ran the proposed algorithm 10 times, each initialized with $k = 30$; the common component was initialized to cover the entire set of data, and the feature saliency values were initialized at 0.5. A local minimum was reached if the change in description length between two iterations was less than 10^{-7}. A typical run of the algorithm is shown in Figure 4.6. In all the ten random runs with this mixture, the four components were always correctly identified. The saliencies of all the ten features, together with their standard deviations (error bars), are shown in Figure 4.7(a). We can conclude that, in this case, the algorithm successfully locates the true clusters and correctly assigns the feature saliencies.

In the second experiment, we considered the Trunk data [122, 252], consisting of two 20-dimensional Gaussians $\mathcal{N}(\mathbf{m}_1, \mathbf{I})$ and $\mathcal{N}(\mathbf{m}_2, \mathbf{I})$, where $\mathbf{m}_1 = (1, \frac{1}{\sqrt{2}}, \ldots, \frac{1}{\sqrt{20}})$, $\mathbf{m}_2 = -\mathbf{m}_1$. Data were obtained by sampling 5000 points from each of these two Gaussians. Note that the features are arranged in descending order of relevance. As above, the stopping threshold was set to 10^{-7} and the initial value of k was set to 30. In all the 10 runs performed, the two components were always detected. The feature saliencies are shown in Figure 4.7(b). The lower the rank number, the more important was the feature. We can see the general trend that as the feature number increases, the saliency decreases, in accordance with the true characteristics of the data.

4.4.2 Real data

We tested our algorithm on several data sets with different characteristics (Table 4.1). The wine recognition data set (`wine`) contains results of chemical analysis of wines grown in different cultivars. The goal is to predict the type of a wine based on its chemical composition; it has 178 data points, 13 features, and 3 classes. The Wisconsin diagnostic breast cancer data set (`wdbc`) was used to obtain a binary diagnosis (benign or malignant) based on 30 features extracted from cell nuclei presented in an image; it has 576 data points. The image segmentation data set (`image`) contains 2320 data points with 19 features from seven classes; each pattern consists of features extracted from a 3×3 region taken from 7 types of outdoor images: brickface, sky, foliage, cement, window, path, and grass. The texture data set (`texture`) consists of 4000 19-dimensional Gabor filter features from a collage of four Brodatz textures [127]. A data set (`zernike`) of 47 Zernike moments extracted from images of handwriting numerals (as in [126]) is also used; there are 200 images for each digit, totaling 2000 patterns. The data sets `wine`, `wdbc`, `image`, and `zernike` are from the UCI machine learning repository (`http://www.ics.uci.edu/~mlearn/MLSummary.html`). This repository has been extensively used in pattern recognition and machine learning studies. Normalization to zero mean and unit variance is performed for all but the texture data set, so as to make the contribution of different features roughly equal a priori. We do not normalize the texture data set because it is already approximately normalized. Since these data sets were collected for supervised classification, the class labels are not involved in our experiment, except for the evaluation of clustering results.

Each data set was first randomly divided into two halves: one for training, another for testing.

Table 4.1: Real world data sets used in the experiment. Each data set has n data points with d features from c classes. One feature with a constant value in `image` is discarded. Normalization is not needed for `texture` because the features have comparable variances.

Abbr.	Full name	n	d	c	Normalized?
`wine`	wine recognition	178	13	3	yes
`wdbc`	Wisconsin diagnostic breast cancer	569	30	2	yes
`image`	image segmentation	2320	18	7	yes
`texture`	Texture data set	4000	19	4	no
`zernike`	Zernike moments of digit images	2000	47	10	yes

Table 4.2: Results of the algorithm over 20 random data splits and algorithm initializations. "Error" corresponds to the mean of the error rates on the testing set when the clustering results are compared with the ground truth labels. $\overline{c}$ denotes the number of Gaussian components estimated. Note that the post-processing does not change the number of Gaussian components. The numbers in parenthesis are the standard deviation of the corresponding quantities.

	Algorithm 4.1		With post-processing	Using all the features	
	error (in %)	$\widehat{c}$	error (in %)	error (in %)	$\widehat{c}$
`wine`	6.61 (3.91)	3.1 (0.31)	6.61 (3.23)	8.06 (3.73)	3 (0)
`wdbc`	9.55 (1.99)	5.65 (0.75)	9.35 (2.07)	10.09 (2.00)	2.70 (0.57)
`image`	20.19 (1.54)	23.1 (1.74)	20.28 (1.60)	32.84 (5.1)	13.8 (1.94)
`texture`	4.04 (0.76)	36.17 (1.19)	4.02 (0.74)	4.85 (0.98)	31.42 (2.81)
`zernike`	52.09 (2.52)	11.3 (0.98)	51.99 (2.32)	56.42 (3.62)	10 (0)

Algorithm 4.1 was run on the training set. The feature saliency values can be refined as described in Section 4.3.4. We evaluated the results by interpreting the fitted Gaussian components as clusters and compared them with the ground truth labels. Each data point in the test set was assigned to the component that most likely generated it, and the pattern was classified to the class represented by the component. We then computed the error rates on the test data. For comparison, we also ran the mixture of Gaussian algorithm in [81] using all the features, with the number of classes of the data set as a lower bound on the number of components. This gives us a fair ground for comparing Gaussian mixtures with and without feature saliency. In order to ensure that we had enough data with respect to the number of features for the algorithm in [81], the covariance matrices of the mixture components were restricted to be diagonal, but were different for different components. The entire procedure was repeated 20 times with different splits of data and initializations of the algorithm. The results are shown in Table 4.2. We also show the feature saliency values of different features in different runs as gray-level image maps in Figure 4.9. For illustrative purpose, we contrast the clusters obtained for the `image` data set with the true class labels in Figure 4.8, after using PCA to project the data into 3D.

From Table 4.2, we can see that the proposed algorithm reduces the error rates when compared with using all the features for all five data sets. The improvement is more significant for the `image` data set, but this may be due to the increased number of components estimated. The high error rate for `zernike` is due to the fact that digit images are inherently more difficult to cluster: for example, "4" can be written in a manner very similar to "9", and it is difficult for any unsupervised learning algorithm to distinguish among them. The post-processing can increase the "contrast" of the feature saliencies, as the image maps in Figure 4.9 show, without deteriorating the accuracy. It is easier to perform "hard" feature selection using these post-processed feature saliencies, if this is required for the application.

4.5 Discussion

4.5.1 Complexity

The major computational load in the proposed algorithm is in the E-step and the M-step. Each E-step iteration computes $O(ndk)$ quantities. As each quantity can be computed in constant time, the time complexity for E-step is also $O(ndk)$. Similarly, the M-step takes $O(ndk)$ time. The total amount of computation depends on the number of iterations required for convergence.

At a first sight, the amount of computation seems to be demanding. However, a close examination reveals that each iteration (E-step and M-step) of the standard EM algorithm also takes $O(ndk)$ time. The value of k in the standard EM, though, is usually smaller, because the proposed algorithm starts with a larger number of components. The number of iterations required for our algorithm is also in general larger because of the increase in the number of parameters. Therefore, it is true that the proposed algorithm takes more time than the standard EM algorithm *with one parameter setting*. However, the proposed algorithm can determine the number of clusters as well as the feature subset. If we want to achieve the same goal with the standard EM algorithm using a wrapper approach, we need to re-run EM multiple times with a different number of components and different feature subsets. The computational demand is much heavier than the proposed algorithm, even with a heuristic search to guide the selection of feature subsets. Another strength of the proposed algorithm is that by initialization with a large number of Gaussian components, the algorithm is less sensitive to the local minimum problem than the standard EM algorithm. We can further reduce the complexity by adopting optimization techniques applicable to standard EM for Gaussian mixture, such as sampling the data, compressing the data [28], or using efficient data structures [203, 224].

For the post-processing step in Section 4.3.4, each computation of the quantity J and its gradient and Hessian takes $O(ndk)$ time. The number of iterations is difficult to predict, as it depends on the optimization routine. However, we can always put an upper bound on the number of iterations and trade speed for the optimality of the results.

4.5.2 Relation to Shrinkage Estimate

One interpretation of Equation (4.6) is that we "regularize" the distribution of each feature in different components by the common distribution. This is analogous to the shrinkage estimator for covariance matrices of class-conditional densities [68], which is a weighted sum of an estimate of the class-specific covariance matrix, and the "global" covariance matrix estimate. In Equation (4.6), the pdf of the l-th feature is also a weighted sum of a component-specific pdf and a common density. An important difference here is that the weight ρ_l is estimated from the data, using the MML principle, instead of being set heuristically, as is commonly done. As shrinkage estimators have found empirical success to combat data scarcity, this "regularization" viewpoint is an alternative explanation for the usefulness of the proposed algorithm.

4.5.3 Limitation of the Proposed Algorithm

A limitation of the proposed algorithm is the feature independence assumption (conditioned on the mixture component). While, empirically, the violation of the independence assumption usually does not affect the accuracy of a classifier (as in supervised learning) or the quality of clusters (as in unsupervised learning), this has some negative influence on the feature selection problem. Specifically, a feature that is redundant because its distribution is independent of the component label given another feature cannot be modelled under the feature independence assumption. As a result, both features are kept. This explains why, in general, the feature saliencies are somewhat high. The post-processing in Section 4.3.4 can cope with this problem because it considers the posterior distribution and therefore can discard features that do not help in identifying the clusters directly.

4.5.4 Extension to Semi-supervised Learning

Sometimes, we may have some knowledge of the class labels of different Gaussian components. This can happen when, say, we adopt a procedure to combine different Gaussian components to form a cluster (*e.g.*, as in [216]), or in a semi-supervised learning scenario, where we can use a small amount of labelled data to help us identify which Gaussian component belongs to which class. This additional information can suggest combination of several Gaussian components to form a single class/cluster, thereby allowing the identification of non-Gaussian clusters. The post-processing step can take advantage of this information.

Suppose we know there are C classes and the posterior probability that pattern $\mathbf{y}_i$ belongs to the c-th class, denoted r_{ic}, can be computed as $r_{ic} = \sum_{j=1}^{k} \beta_{cj} P(z_i = j|\mathbf{y}_i)$. For example, if we know that the components 4, 6, and 10 are from class 2, we can set $\beta_{2,4} = \beta_{2,6} = \beta_{2,10} = 1/3$ and the other $\beta_{2,j}$ to be zero. The post-processing is modified accordingly: redefine t_i in Equation (4.31) to $t_i = \arg\max_c r_{ic}$, *i.e.*, it becomes the class label for $\mathbf{y}_i$ in view of the extra information; replace $\log P(z_i = t_i|\mathbf{y}_i)$ in Equation (4.31) by $\log r_{i,t_i}$. The gradient and Hessian can still be computed easily after noting that

$$\begin{aligned} \frac{\partial w_{ij}}{\partial \rho_l} &= w_{ij} \frac{\partial}{\partial \rho_l} \log w_{ij} = w_{ij}(h_{ilj} - g_{il}) \\ \frac{\partial}{\partial \rho_l} \log r_{ic} &= \frac{1}{r_{ic}} \sum_{j=1}^{k} \beta_{cj} \frac{\partial}{\partial \rho_l} w_{ij} = \sum_{j=1}^{k} \frac{\beta_{cj} w_{ij}}{r_{ic}} h_{ilj} - g_{il}. \end{aligned} \tag{4.32}$$

We can then optimize the modified J in Equation (4.31) to carry out the post-processing.

4.5.5 A Note on Maximizing the Posterior Probability

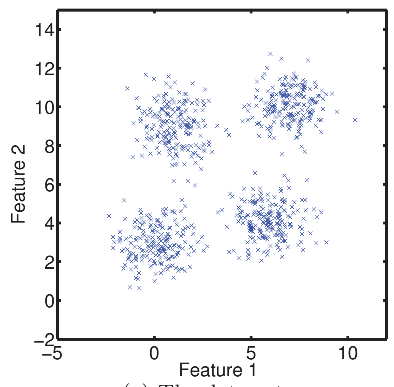
The sum of the logarithm of the maximum posterior probability considered in the post-processing in Section 4.3.4 can be regarded as the sample estimate of an unorthodox type of entropy (see [141]) for the posterior distribution. It can be regarded as the limit of Renyi's entropy $R_\alpha(P)$ when α tends to infinity, where

$$R_\alpha(P) = \frac{1}{1-\alpha} \log \sum_{j=1}^{k} p_i^\alpha. \tag{4.33}$$

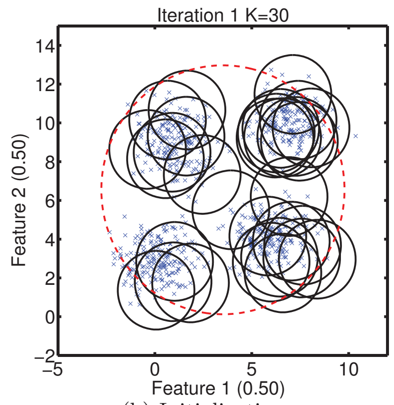
When this entropy is used for parameter estimation under the maximum entropy framework, the corresponding procedure is closely related to minimax inference. Other functions on the posterior probabilities can also be used, such as the Shannon entropy of the posterior distribution. Preliminary study shows that the use of different types of entropy does not affect the results significantly.

4.6 Summary

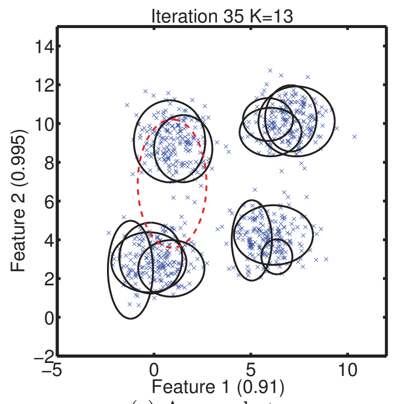
Given n points in d dimension, we have presented an EM algorithm to estimate the saliencies of individual features and the best number of components for Gaussian-mixture clustering. The proposed algorithm can avoid running EM many times with different numbers of components and different feature subsets, and can achieve better performance than using all the available features for clustering. By initializing with a large number of mixture components, our EM algorithm is less prone to the problem of poor local minima. The usefulness of the algorithm was demonstrated on both synthetic and benchmark real data sets.



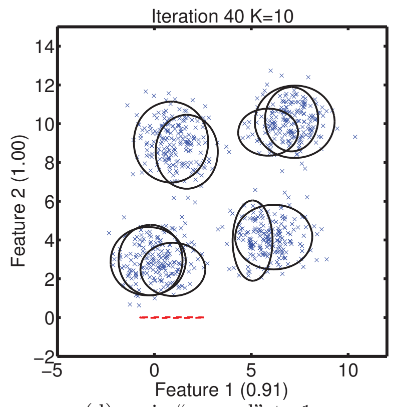
(a) The data set



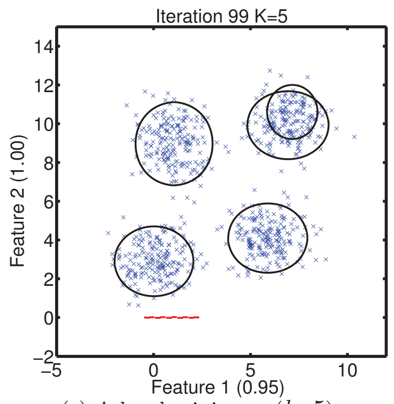
(b) Initialization



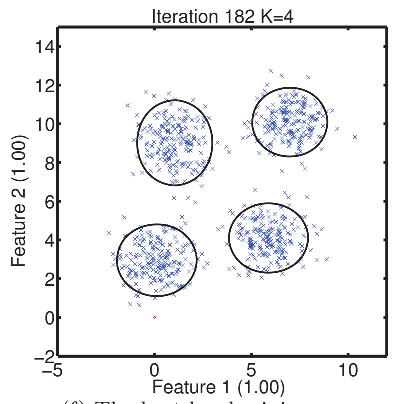
(c) A snapshot



(d) ρ_2 is "pruned" to 1

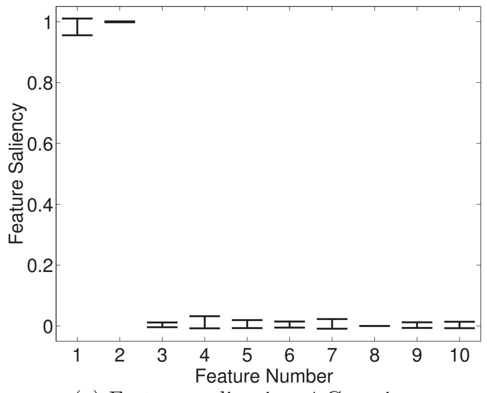


(e) A local minimum (k=5)

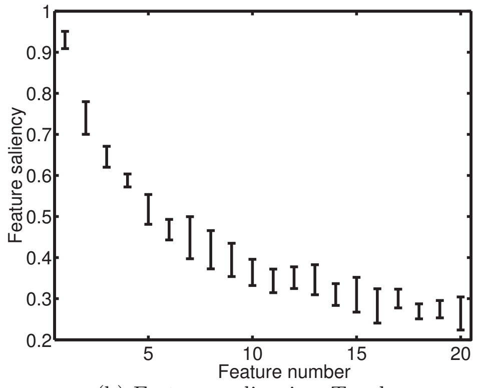


(f) The best local minimum

Figure 4.6: An example execution of the proposed algorithm. The solid ellipses represent the Gaussian mixture components; the dotted ellipse represents the common density. The number in parenthesis along the axis label is the feature saliency; when it reaches 1, the common component is no longer applicable to that feature. Thus, in (d), the common component degenerates to a line; when the feature saliency for feature 1 also becomes 1, as in Figure 4.6(f), the common density degenerates to a point at $(0, 0)$.

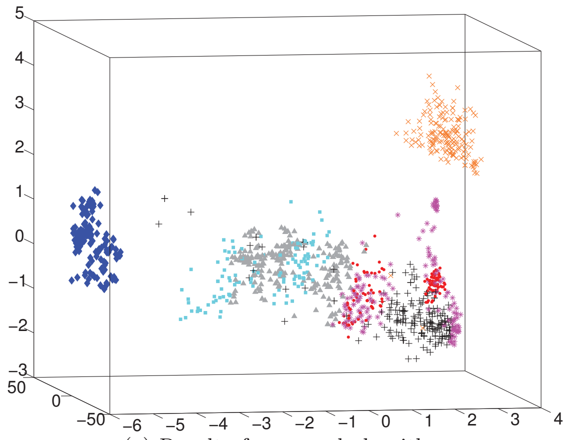


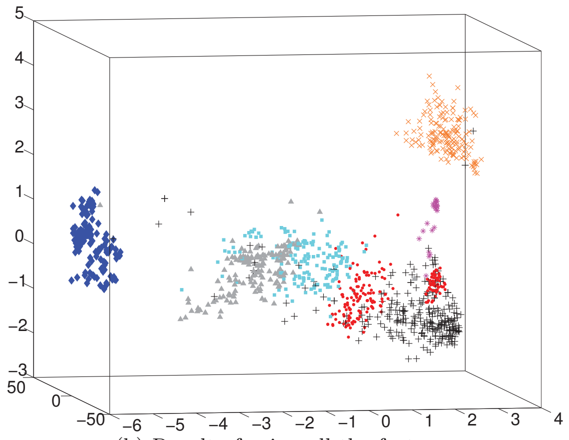
(a) Features saliencies: 4 Gaussian



(b) Features saliencies: Trunk

Figure 4.7: Feature saliencies for (a) the 10-D 4 Gaussian data set used in Figure 4.6(a), and (b) the Trunk data set. The mean values plus and minus one standard deviation over ten runs are shown. Recall that features 3 to 10 for the 4 Gaussian data set are the noisy features.


(a) Result of proposed algorithm


(b) Result of using all the features

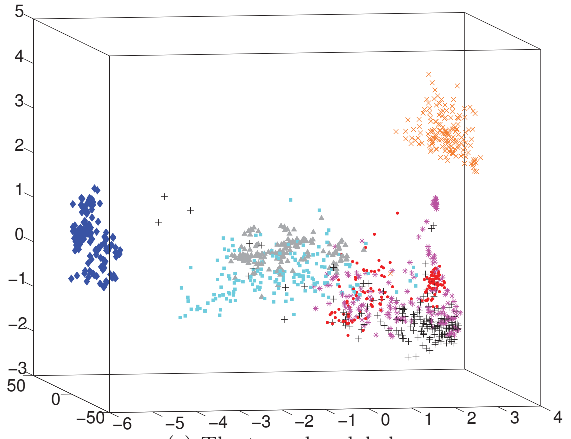
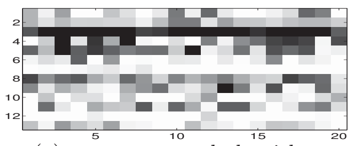
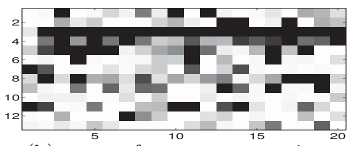
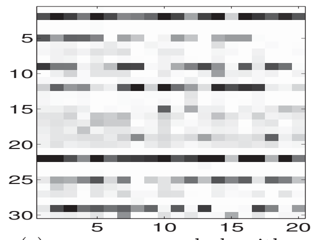
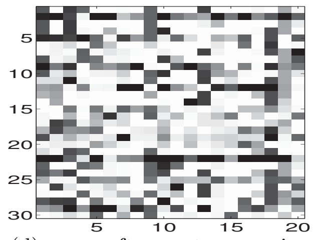

(c) The true class labels

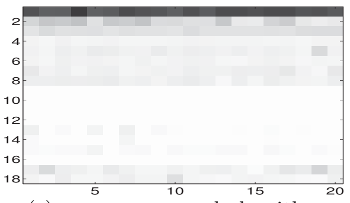
Figure 4.8: A figure showing the clustering result on the `image` data set. Only the labels for the testing data are shown. (a) The true class labels. (b) The clustering results by Algorithm 4.1. (c) The clustering result using all the features. The data points are reduced to 3D by PCA. A cluster is matched to its majority class before plotting. The error rates for the proposed algorithm and the algorithm using all the features in this particular run are 22% and 30%, respectively.

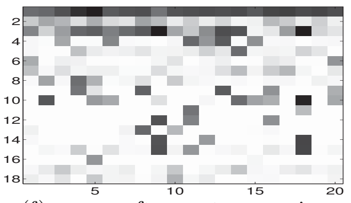

(a) `wine`, proposed algorithm

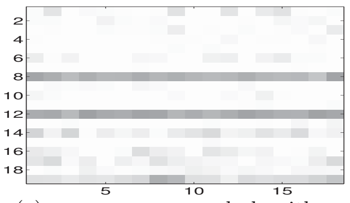

(b) `wine`, after post-processing

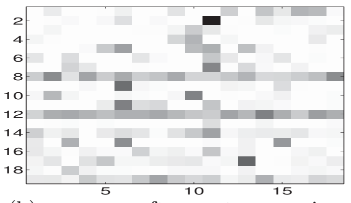

(c) `wdbc`, proposed algorithm

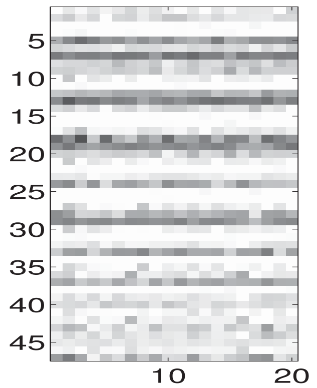

(d) `wdbc`, after post-processing


(e) `image`, proposed algorithm


(f) `image`, after post-processing


(g) `texture`, proposed algorithm


(h) `texture`, after post-processing


(i) `zernike`, proposed algorithm

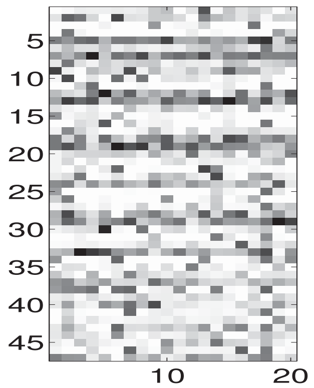

(j) `zernike`, after post-processing

Figure 4.9: Image maps of feature saliency for different data sets with and without the post-processing procedure. Feature saliency of 1 (0) is shown as a pixel of gray level 255 (0). The vertical and horizontal axes correspond to the feature number and the trial number, respectively.

Chapter 5

Clustering With Constraints

In Section 1.4, we introduced instance-level constraints as a type of side-information for clustering. In this chapter, we shall examine the drawbacks of the existing clustering under constraints algorithms, and propose a new algorithm that can remedy the defects.

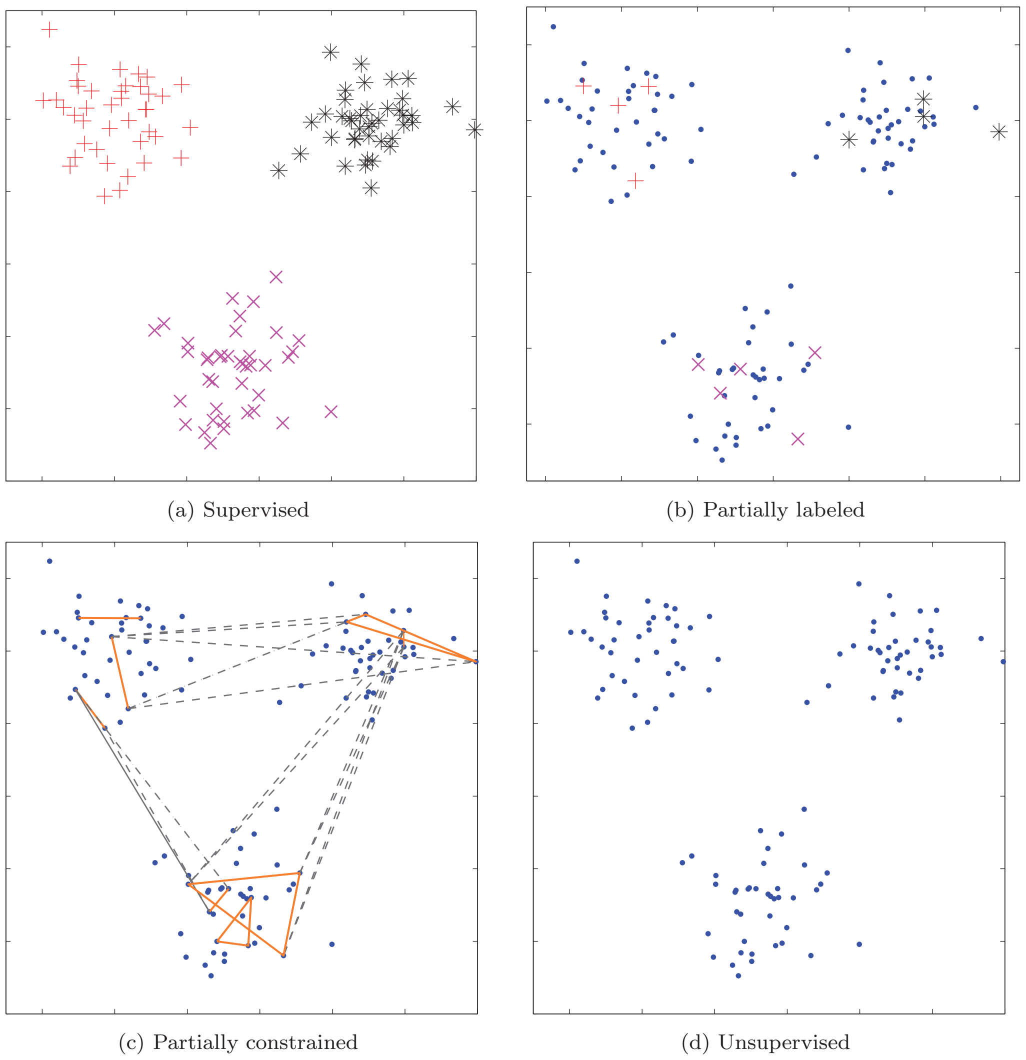
Recall that there are two types of instance-level constraints: a must-link/positive constraint requires two or more objects to be put in the same cluster, whereas a must-not-link/negative constraint requires two or more objects to be placed in different clusters. Often, the constraints are pairwise, though one can extend them to multiple objects [231, 167]. Constraints are particularly appropriate in a clustering scenario, because there is no clear notion of the target classes. On the other hand, the user can suggest if two or more objects should be included in the same cluster or not. This can be done in an interactive manner, if desired. Side-information can improve the robustness of a clustering algorithm towards model mismatch, because it provides additional clues for the desirable clusters other than the shape of the clusters, as suggested by the parametric model. Side-information has also been found to alleviate the problem of local minima of the clustering objective function.

Clustering with instance-level constraints is different from learning with partially-labeled data, also known as transductive learning or semi-supervised learning [136, 288, 157, 169, 289, 287, 98, 195], where the class labels of some of the objects are provided. Constraints only reveal the relationship among the labels, not the labels themselves. Indeed, if the "absolute" labels can be specified, the user is no longer facing a clustering task, and a supervised method should be adopted instead.

We contrast different learning settings according to the type of information available in Figure 5.1. At one end of the spectrum, we have supervised learning (Figure 5.1(a)), where the labels of all the objects are known. At the other end of the spectrum, we have unsupervised learning (Figure 5.1(d)), where the label information is absent. In between, we can have partially labeled data (Figure 5.1(b)), where the true class labels of some of the objects are known. The main scenario considered in this paper is depicted in Figure 5.1(c): there is no label information, but must-link and must-not-link constraints (represented by solid and dashed lines, respectively) are provided. Note that the settings exemplified in Figures 5.1(a) and 5.1(b) are classification-oriented because there is a clear definition of different classes. On the other hand, the setups in Figures 5.1(c) and 5.1(d) are clustering-oriented, because no precise definitions of classes are given. The clustering algorithm needs to discover the classes.

5.0.1 Related Work

Different algorithms have been proposed for clustering under instance-level constraints. In [262], the four primary operators in COBWEB were modified in view of the constraints. The k-means


(a) Supervised (b) Partially labeled

(c) Partially constrained (d) Unsupervised

Figure 5.1: Supervised, unsupervised, and intermediate. In this figure, dots correspond to points without any labels. Points with labels are denoted by circles, asterisks and crosses. In (c), the must-link and must-not-link constraints are denoted by solid and dashed lines, respectively.

algorithm was modified in [263] to avoid violating the constraints when different objects are assigned to different clusters. However, the algorithm can fail even when a solution exists. Positive constraints served as "short-cuts" in [148] to modify the dissimilarity measure for complete-link clustering. There can be catastrophic consequences if a single constraint is incorrect, because the dissimilarity matrix can be greatly distorted by a wrong constraint. Spectral clustering was modified in [138] to work with constraints, which augmented the affinity matrix. Constraints were incorporated into image segmentation algorithms by solving the constrained version of the corresponding normalized cut problem, with smoothness of cluster labels explicitly incorporated in the formulation [279]. Hidden Markov random field was used in [14] for k-means clustering with constraints. Constraints have also been used for metric-learning [274]; in fact, the problems of metric-learning and k-means clustering with constraints were considered simultaneously in [21]. Because the problem of k-means with metric-learning is related to EM clustering with a common covariance matrix, the work in [21] may be viewed as related to EM clustering with constraints. The work in [158] extended the work in [21]

Summary	Key ideas	Examples
Distance editing	Modify the distance/proximity matrix due to the constraints	[148, 138]
Constraints on labels	The cluster labels are inferred under the restriction that the constraints are always satisfied	[262, 263, 279]
Hidden Markov random field	Cluster labels constitute a hidden Markov random field; feature vectors are assumed to be independent of each other given cluster labels	[14, 21, 12, 158, 176, 161, 286]
Modify generation model	Generation process of data points that participate in constraints is modified	[231, 166, 167]
Constraints resolution	Clustering solution is obtained by resolving constraints only	[10]

Table 5.1: Different algorithms for clustering with constraints.

by studying the relationship between constraints and the kernel k-means algorithms. Ideas based on hidden Markov random field have also been used for model-based clustering with constraints [14, 176, 161]; the difference between these three methods lies in how the inference is conducted. In particular, the method in [14] used iterative conditional mode (ICM), the method in [176] used Gibbs sampling, and the method in [161] used a mean-field approximation. The approach in [286] is similar to [161], since both used mean-field approximation. However, the authors of [286] also considered the case when each class is modeled by more than one component. A related idea was presented in [231], which uses a graphical model for generating the data with constraints. A fairly different route to clustering under constraints was taken by the authors in [10] under the name "correlation clustering", which used only the positive and negative constraints (and no information on the objects) for clustering. The number of clusters can be determined by the constraints.

Table 5.1 provides a summary of these algorithms for clustering under constraints. In most of these approaches, clustering with constraints has been shown to improve the quality of clustering in different domains. Example applications include text classification [14], image segmentation [161], and video retrieval [231].

5.0.2 The Hypothesis Space

An important issue in parametric clustering under constraints, namely the hypothesis space, has virtually been ignored in the current literature. Here, we adopt the terminology from inductive learning and regard "hypothesis space" as the space of all possible solutions to the clustering task. Since partitional clustering can be viewed as the construction of a mapping from a set of objects to a set of cluster labels, its hypothesis space is the set of all possible mappings between the objects (or their representations) and the cluster labels. In a non-parametric clustering algorithm such as pairwise clustering [114] and methods based on graph-cut [234, 272], there is no restriction on this hypothesis space. A particular non-parametric clustering algorithm selects the best clustering solution in the space according to some criterion function. In other words, if a poor criterion function is used (perhaps due to the influence of constraints), one can obtain a counter-intuitive clustering solution such as the one in Figure 5.3(c), where very similar objects can be assigned different cluster labels. Note that objects in non-parametric clustering, unlike in parametric clustering, may not have a feature vector representation. They can be represented, for example, by pairwise affinity or dissimilarity measure with higher order [1].

The hypothesis space in parametric clustering is typically much smaller, because the parametric assumption imposes restrictions on the cluster boundaries. While these restrictions are generally

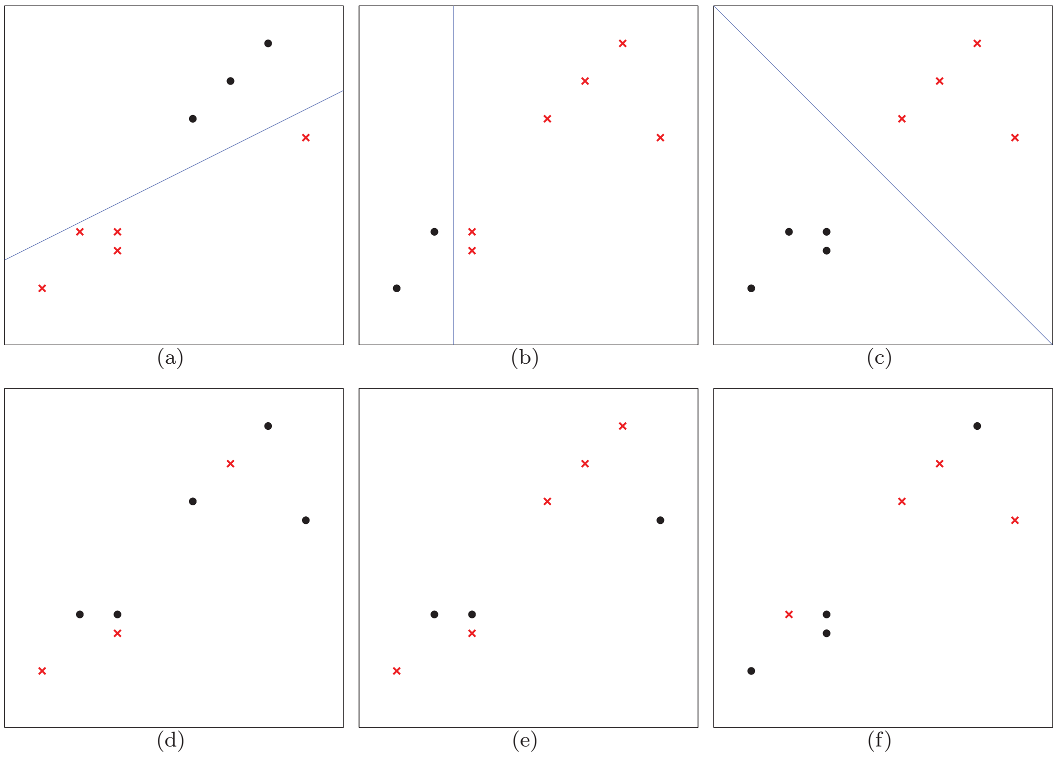


Figure 5.2: An example contrasting parametric and non-parametric clustering. The particular parametric family considered here is a mixture of Gaussian with a common covariance matrix. This is reflected by the linear cluster boundary. The clustering solutions in (a) to (c) are in the hypothesis space induced by this model assumption, and the clustering solutions in (d) to (f) are outside the hypothesis space, and thus can never be obtained, no matter which objective function is used. On the other hand, all of these six solutions are within the hypothesis space of non-parametric clustering. It is possible that the clustering solutions depicted in (d), (e), and (f) may be obtained if a poor clustering objective function is used.

perceived as a drawback, they become advantageous when they prevent counter-intuitive clustering solutions such as the one in Figure 5.3(c) from appearing. These clustering solutions are simply outside the hypothesis space of parametric clustering, and are never attainable irrespective of how the constraints modify the clustering objective function.

An example contrasting parametric and non-parametric clustering is shown in Figure 5.2. The particular parametric family considered in this example is a Gaussian distribution with common covariance matrix, resulting in linear cluster boundaries.

5.0.2.1 Inconsistent Hypothesis Space in Existing Approaches

The basic idea of most of the existing parametric clustering with instance-level constraints algorithms [263, 14, 21, 12, 158, 176, 161, 286] is to use some variants of hidden Markov random fields to model the cluster labels and the feature vectors. Given the cluster label of the object, its feature vector is assumed to be independent of the feature vectors and the cluster labels of all the other objects. The cluster labels, which are hidden (unknown), form a Markov random field, with the potential function in this random field related to the satisfiability of the constraints based on the cluster labels.

There is an unfortunate consequence of adopting the hidden Markov random field, however. For objects participating in the constraints, their cluster labels are determined by the cluster parameters, associated feature vectors and the constraints. On the other hand, for data points without constraints, the cluster labels are determined by only the cluster parameters and associated feature vectors. We can thus see that there is an inconsistency in how the objects obtain their cluster labels. In other words, two identical objects, one with a constraint and one without, can be assigned different cluster labels! This is the underlying reason for the problem illustrated in Figure 5.3(d), where two objects with almost identical feature vectors are assigned different labels due to the constraints.

From a generative viewpoint, the above inconsistency is caused by the difference in how data points with and without constraints are generated. For the data points without constraint, each of them is generated in an identical and independent manner according to the current cluster parameter value. On the other, all the data points with constraints are generated simultaneously by first choosing the cluster labels according to the hidden Markov random field, followed by the generation of the feature vectors based on the cluster labels. It is a dubious modeling assumption that "posterior" knowledge such as the set of instance-level constraints, which are solicited from the user after observing the data, should control how the data are generated in the first place.

Note that this inconsistency does not exist if all the objects to be clustered are involved in some constraints determined by the properties of the objects. This is commonly encountered in image segmentation [128], where pixel attributes (e.g. intensities or filter responses) and spatial coherency based on the locations of the pixels are considered simultaneously to decide the segment label. In this case, the cluster labels of all the objects are determined by both the constraints and the feature vectors.

5.0.2.2 Proposed Solution

We propose to eliminate the problem of inconsistent hypothesis space by enforcing a uniform way to determine the cluster label of an object. We use the same hypothesis space of standard parametric clustering for parametric clustering under constraints. The constraints are only used to bias the search of a clustering solution within this hypothesis space. Since each clustering solution in this hypothesis space can be represented by the cluster parameters, the constraints play no role in determining the cluster labels, given the cluster parameters. The quality of the cluster parameters with respect to the constraints is computed by examining how well the cluster labels (determined by the cluster parameters) satisfy the constraints. However, cluster parameters that fit the constraints well may not fit the data well. We need a tradeoff between these two goals. This can be done by maximizing a weighted sum of the data log-likelihood and a constraint fit term. The details will be presented in Section 5.3.

5.1 Preliminaries

Given a set of n objects $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$, (probabilistic) parametric partitional clustering discovers the cluster structure of the data under the assumption that data in a cluster are generated according to a certain probabilistic model $p(\mathbf{y}|\theta_j)$, with θ_j representing the parameter vector for the j-th cluster. For simplicity, the number of clusters, k, is assumed to be specified by the user, though model selection strategy (such as minimum description length [81] and stability [162]) can be applied to determine k, if desired. The distribution of the data can be written as a finite mixture distribution, i.e.,

$$p(\mathbf{y}) = \sum_z p(\mathbf{y}|z)p(z) = \sum_{j=1}^{k} \alpha_j p(\mathbf{y}|\theta_j). \tag{5.1}$$

Here, z denotes the cluster label, α_j denotes $p(z = j)$ (the prior probability of cluster j), and $p(\mathbf{y}|\theta_j)$ corresponds to $p(\mathbf{y}|z = j)$. Clustering is performed by estimating the model parameter θ, defined by $\theta = (\alpha_1, \dots, \alpha_k, \theta_1, \dots, \theta_k)$. By applying the maximum likelihood principle, θ can be estimated as $\theta = \arg\max_{\theta'} \mathcal{L}(\theta'; \mathcal{Y})$, where the log-likelihood $\mathcal{L}(\theta; \mathcal{Y})$ is defined as

$$\mathcal{L}(\theta; \mathcal{Y}) = \sum_{i=1}^{n} \log p(\mathbf{y}_i) = \sum_{i=1}^{n} \log \sum_{j=1}^{k} \alpha_j p(\mathbf{y}|\theta_j). \tag{5.2}$$

This maximization is often done by the EM algorithm [58] by regarding z_i (the cluster label of $\mathbf{y}_i$) as the missing data. The posterior probability $p(z = j|\mathbf{y})$ represents how likely it is that $\mathbf{y}$ belongs to the j-th cluster. If a hard cluster assignment is desired, the MAP (maximum a posteriori) rule can be applied based on the model in Equation (5.1), i.e., the object $\mathbf{y}$ is assigned to the j-th cluster $(z = j)$ if

$$j = \arg\max_{l} \frac{\alpha_l p(\mathbf{y}|\theta_l)}{\sum_{l'} \alpha_l p(\mathbf{y}|\theta_l)}. \tag{5.3}$$

5.1.1 Exponential Family

While there are many possibilities for the form of the probability distribution $p(y|\theta_j)$, it is very common to assume that $p(y|\theta_j)$ belongs to the exponential family. The distribution $p(\mathbf{y}|\theta_j)$ is in the exponential family if it satisfies the following two criteria: the support of $p(\mathbf{y}|\theta_j)$ (the set of $\mathbf{y}$ with non-zero probability) is independent of the value of θ_j, and that $p(\mathbf{y}|\theta_j)$ can be written in the form

$$p(\mathbf{y}|\theta_j) = \exp\left(\phi(\mathbf{y})^T \psi(\theta_j) - A(\theta_j)\right). \tag{5.4}$$

Here, $\phi(\mathbf{y})$ transforms the data $\mathbf{y}$ to become the "sufficient statistics", meaning that $\phi(\mathbf{y})$ encompasses all the relevant information of $\mathbf{y}$ in the computation of $p(\mathbf{y}|\theta_j)$. The function $A(\theta_j)$, also known as the log-partition function, normalizes the density so that it integrates to one over all $\mathbf{y}$. The function $\psi(\theta_j)$ transforms the parameter and enables us to adopt different parameterizations of the same density. When $\psi(.)$ is the identity mapping, the density is said to be in natural parameterization, and θ_j is known as the natural parameter of the distribution. The function $A(\theta_j)$ then becomes the cumulant generating function, and the derivative of $A(\theta_j)$ generates the cumulant of the sufficient statistics. For example, the gradient and Hessian of $A(\theta_j)$ (with respect to θ_j) lead to the expected value and the covariance matrix for the sufficient statistics, respectively. Note that $A(\theta_j)$ is a convex function, and the domain of θ_j where the density is well-defined under natural parameterization is also convex.

As an example, consider a multivariate Gaussian density with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Its pdf is given by

$$p(\mathbf{y}) = \exp\left(-\frac{d}{2}\log(2\pi) + \frac{1}{2}\log\det\boldsymbol{\Sigma}^{-1} - \frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{y}-\boldsymbol{\mu})\right), \tag{5.5}$$

where d is the dimension of the feature vector $\mathbf{y}$. If we define $\boldsymbol{\Upsilon} = \boldsymbol{\Sigma}^{-1}$ and $\boldsymbol{\nu} = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$, the above can be rewritten as

$$p(\mathbf{y}) = \exp\left(\operatorname{trace}\left(-\frac{1}{2}\mathbf{y}\mathbf{y}^T\boldsymbol{\Upsilon}\right) + \mathbf{y}^T\boldsymbol{\nu} - \frac{d}{2}\log(2\pi) + \frac{1}{2}\log\det\boldsymbol{\Upsilon} - \frac{1}{2}\boldsymbol{\nu}^T\boldsymbol{\Upsilon}^{-1}\boldsymbol{\nu}\right). \tag{5.6}$$

From this, we can see that the sufficient statistics consist of $-\frac{1}{2}\mathbf{y}\mathbf{y}^T$ and $\mathbf{y}$. The set of natural parameter is given by $(\boldsymbol{\Upsilon}, \boldsymbol{\nu})$. The parameter $\boldsymbol{\nu}$ can take any value in $\mathbb{R}^d$, whereas $\boldsymbol{\Upsilon}$ can only assume values in the positive-definite cone of d by d symmetric matrices. Both these two sets are convex, as expected. The log-cumulant function is given by

$$A(\theta) = \frac{d}{2}\log(2\pi) - \frac{1}{2}\log\det\boldsymbol{\Upsilon} + \frac{1}{2}\boldsymbol{\nu}\boldsymbol{\Upsilon}^{-1}\boldsymbol{\nu}, \tag{5.7}$$

which can be shown to be convex within the domain of $\boldsymbol{\Upsilon}$ and $\boldsymbol{\nu}$, where the density is well-defined.

It is interesting to note that the exponential family is closely related to Bregman divergence [27]. For any Bregman divergence $D_\rho(.,.)$ derived from a strictly convex function $\rho(.)$, one can construct a function f_ρ such that

$$p(\mathbf{y}) = \exp\left(-D_\rho(\mathbf{y}, \boldsymbol{\mu})\right) f_\rho(\mathbf{y})$$

is a member of the exponential family. Here, $\boldsymbol{\mu}$ is the moment parameter, meaning that it is the expected value of the sufficient statistics[1]. The cumulant generating function of the density is given by the Legendre dual of $\rho(.)$. One important consequence of this relationship is that soft-clustering (clustering where an object can be partially assigned to a cluster) based on any Bregman divergence can be done by fitting a mixture of the corresponding distribution in the exponential family, as argued in [9]. Since Bregman divergence includes many useful distance measures[2] as special cases (such as Euclidean distance and Kullback-Leibler divergence, and see [9] for more), a mixture density, with each component density in the exponential family, covers many interesting clustering scenarios.

5.1.2 Instance-level Constraints

We assume that the user has provided side-information in the form of a set of instance-level constraints (denoted by $\mathcal{C}$). The set of must-link constraints, denoted by $\mathcal{C}^+$, is represented by the indicator variables $\tilde{a}_{hi}$, such that $\tilde{a}_{hi} = 1$ iff $\mathbf{y}_i$ participates in the h-th must-link constraint. For example, if the user wants to state that the pair $(\mathbf{y}_2, \mathbf{y}_8)$ participates in the fifth must-link constraint, the user sets $\tilde{a}_{5,2} = 1$, $\tilde{a}_{5,8} = 1$, and $\tilde{a}_{5,i} = 0$ for all other i. This formulation, while less explicit than the formulation in [161], which specifies the pairs of points participating in the constraints directly, allows easy generalization to group constraints [166]: we simply set $\tilde{a}_{hi}$ to one for all $\mathbf{y}_i$ that are involved in the h-th group constraint. We also define $a_{hi} = \tilde{a}_{hi}/\sum_i \tilde{a}_{hi}$, where a_{hi} can be perceived as the "normalized" indicator matrix, in the sense that $\sum_i a_{hi} = 1$. The set of must-not-link constraints, denoted by $\mathcal{C}^-$, is represented similarly by the variables $\tilde{b}_{hi}$ and b_{hi}. Specifically, $\tilde{b}_{hi} = 1$ if $\mathbf{y}_i$ participates in the h-th must-not-link constraint, and $b_{hi} = \tilde{b}_{hi}/\sum_i \tilde{b}_{hi}$. Note that $\{a_{hi}\}$ and $\{b_{hi}\}$ are highly sparse, because each constraint provided by the user involves only a small number of points (two if all the constraints are pairwise).

5.2 An Illustrative Example

In this section, we describe a simple example to illustrate an important shortcoming of parametric clustering under constraints methods based on hidden Markov random field – the approach common in the literature [263, 14, 21, 12, 158, 176, 161, 286]. In Figure 5.3, there are altogether 400 data points generated by four different Gaussian distributions. The task is to split this data into two clusters.

[1]The strict convexity of $A(.)$ implies that there is an one-to-one correspondence between moment parameter and natural parameter. While the existence of such a mapping is easy to show, constructing such a mapping can be difficult in general.

[2]Strictly speaking, Bregman divergence can be asymmetric and hence is not really a distance function.

Suppose the user, perhaps due to domain knowledge, prefers a “left” and a “right” cluster (as shown in Figure 5.3(c)) to the more natural solution of a “top” and a “bottom” cluster (as shown in Figure 5.3(b)). This preference can be expressed via the introduction of two must-link constraints, represented by the solid lines in Figure 5.3(a). When we apply an algorithm based on hidden Markov random field to discover the two clusters in this example, we can get a solution shown in Figure 5.3(c). While cluster labels of the points involved in the constraints are modified by the constraints, there is virtually no difference in the resulting cluster structure when compared with the natural solution in Figure 5.3(b). This is because the change in the cluster labels of the small number of points in constraints does not significantly affect the cluster parameters. Not only are the clusters not what the user seeks, but also the clustering solution is counter-intuitive: the cluster labels of points involved in the constraints are different from their neighbors (see the big cross and plus in Figure 5.3(c); the symbols are enlarged for clarity).

Similar phenomena of “non-smooth” clustering solution have been observed in [279] in the context of normalized cut clustering with constraints. A variation of the same problem has been used as a motivation for the “space-level” instead of “instance-level” constraints in [148]. One way to understand the cause of this problem is that the use of hidden Markov random field effectively puts an upper bound on the maximum influence of a constraint, irrespective of how large the penalty for constraint violation is. So, the adjustment of the tradeoff parameters cannot circumvent this problem. Since this problem is not caused by the violation of any constraints, the inclusion of negative constraints cannot help.

5.2.1 An Explanation of The Anomaly

In order to have a better understanding of why an “unnatural” solution depicted in Figure 5.3(d) is obtained, let us examine the hidden Markov random field approach for clustering under constraints in more detail. In this approach, the distribution of the cluster labels (represented by z_i) and the feature vectors (represented by $\mathbf{y}_i$) can be written as

$$p(\mathbf{y}_1, \ldots, \mathbf{y}_n | z_1, \ldots, z_n, \theta) = \prod_{i=1} p(\mathbf{y}_i | z_i)$$
$$p(z_1, \ldots, z_n) \propto \exp\bigl(-\mathcal{H}(z_1, \ldots, z_n, \mathcal{C}^+, \mathcal{C}^-)\bigr).$$

One typical choice of the potential function $\mathcal{H}(z_1, \ldots, z_n, \mathcal{C}^+, \mathcal{C}^-)$ of the cluster labels is to count the number of constraint violations:

$$\mathcal{H}(z_1, \ldots, z_n, \mathcal{C}^+, \mathcal{C}^-) = \lambda^+ \sum_{(i,j)\in\mathcal{C}^+} I(z_i \neq z_j) + \lambda^- \sum_{(i,j)\in\mathcal{C}^-} I(z_i = z_j), \tag{5.8}$$

where λ^+ and λ^- are the penalty parameters for the violation of the must-link and must-not-link constraints, respectively. This potential function can be derived [161] by the maximum entropy principle, with constraints (as in constrained optimization) on the number of violations of the two types of instance-level constraints. The assignment of points to different clusters is determined by the posterior probability $p(z_1, \ldots, z_n | \mathbf{y}_1, \ldots, \mathbf{y}_n, \theta)$. Clustering is performed by searching for the

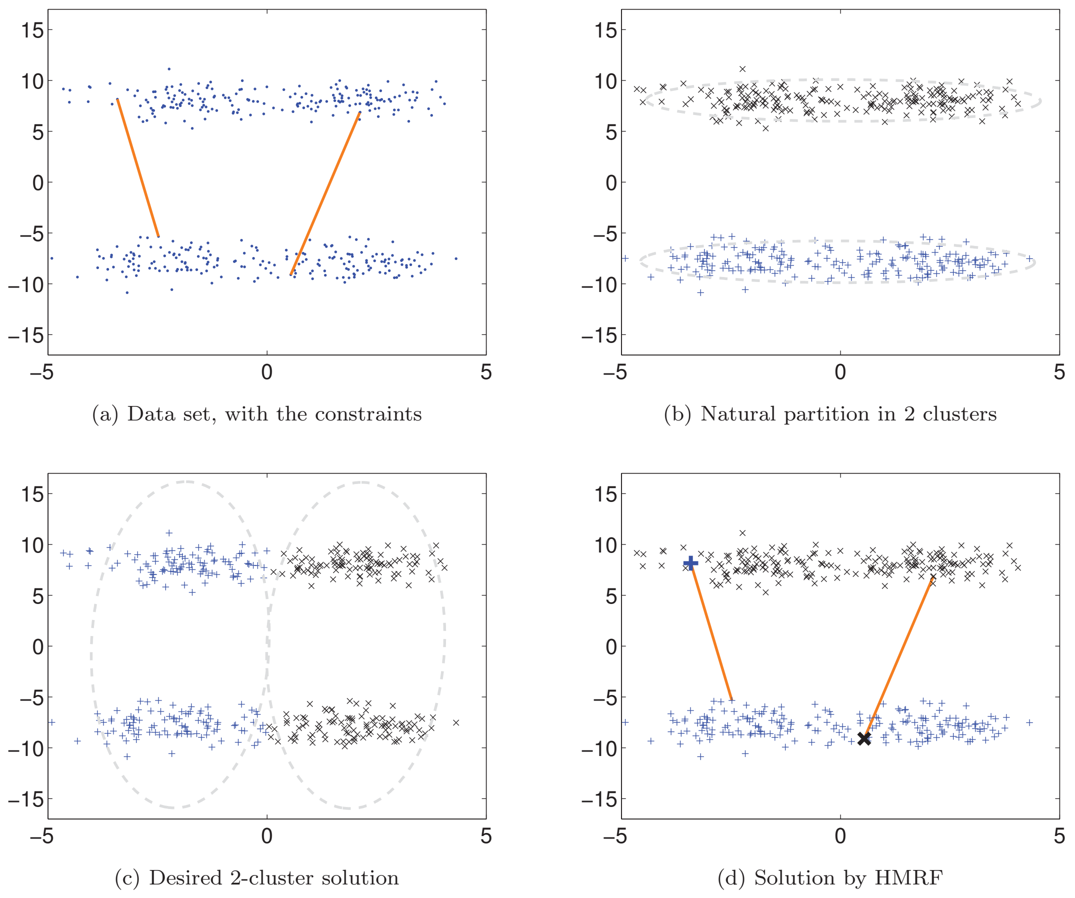

(a) Data set, with the constraints (b) Natural partition in 2 clusters (c) Desired 2-cluster solution (d) Solution by HMRF

Figure 5.3: A simple example of clustering under constraints that illustrates the limitation of hidden Markov random field (HMRF) based approaches.

parameters that maximize the log-likelihood $p(\mathbf{y}_1, \ldots, \mathbf{y}_n|\theta)$. Because

$$\begin{aligned} p(\mathbf{y}_1, \ldots, \mathbf{y}_n|\theta) &= \sum_{z_1,\ldots,z_n} p(\mathbf{y}_1, \ldots, \mathbf{y}_n|z_1, \ldots, z_n, \theta)p(z_1, \ldots, z_n|\theta) \\ &\approx \arg \max_{z_1,\ldots,z_n} p(\mathbf{y}_1, \ldots, \mathbf{y}_n|z_1, \ldots, z_n, \theta)p(z_1, \ldots, z_n|\theta), \end{aligned} \tag{5.9}$$

the result of maximizing $p(\mathbf{y}_1, \ldots, \mathbf{y}_n|\theta)$ is often similar to the result of maximizing the "hard assignment log-likelihood", defined by $\arg \max_{z_1,\ldots,z_n} p(\mathbf{y}_1, \ldots, \mathbf{y}_n|z_1, \ldots, z_n, \theta)p(z_1, \ldots, z_n|\theta)$. This illustrates the relationship between "hard" clustering under constraints approaches (such as in [263]) and the "soft" approaches (such as in [161] and [14]).

For ease of illustration, assume that $p(\mathbf{y}|z = j)$ is a Gaussian with mean vector $\boldsymbol{\mu}_j$ and identity covariance matrix. The maximization of $p(\mathbf{y}_1, \ldots, \mathbf{y}_n|z_1, \ldots, z_n, \theta)p(z_1, \ldots, z_n|\theta)$ for the clustering under constraints example in Figure 5.3 is equivalent to the minimization of

$$\sum_{i=1}^{n} \sum_{j=1}^{2} I(z_i = j)||\mathbf{y}_i - \boldsymbol{\mu}_j||^2 + \lambda^+ \sum_{(i,j)\in\mathcal{C}^+} I(z_i \neq z_j),$$

where the potential function of the Markov random field is as defined in Equation (5.8), and $\mathcal{C}^+$ contains the two must-link constraints. Note that the first term, the sum of square Euclidean distances between data points and the corresponding cluster centers, is the cost function for standard k-means clustering.

We are going to compare two cluster configurations. The configuration "LR", which consists of a "left" and a "right" cluster, can be represented by $\boldsymbol{\mu}_1^{\mathrm{LR}} = (-2, 0)$ and $\boldsymbol{\mu}_2^{\mathrm{LR}} = (2, 0)$, and this corresponds to the partition sought by the user in Figure 5.3(c). The configuration "TB", which consists of a "top" and a "bottom" cluster, can be represented by $\boldsymbol{\mu}_1^{\mathrm{TB}} = (0, -8)$ and $\boldsymbol{\mu}_2^{\mathrm{TB}} = (0, 8)$, and this corresponds to the "natural" solution shown in Figure 5.3(b). When λ^+ is very small, the natural solution "TB" is preferable to "LR", because the points, on average, are closer to the cluster centers in "TB", and the penalty for constraint violation is negligible. As λ^+ increases, the cost for selecting "TB" increases. When $(\lambda^+ + ||\mathbf{y}_i - \boldsymbol{\mu}_2^{\mathrm{TB}}||^2) > ||\mathbf{y}_i - \boldsymbol{\mu}_1^{\mathrm{TB}}||^2$, ($\mathbf{y}_i$ is the point under constraint in the upper left point clouds), switching the cluster label of $\mathbf{y}_i$ from "x" to "+" leads to a lower cost for the "TB" configuration. This switching of cluster label affects the cluster centers in the "TB" configuration. However, its influence is minimal because there is only one such point, and the sum of the square error term in the objective function is dominated by the remaining points that are not involved in constraints. As a result, the sum of square term is minimized when the cluster centers are effectively unmodified from the "TB" configuration. This leads to the counter-intuitive clustering solution in Figure 5.3(c), where the constraints are satisfied, but the cluster labels are "discontinuous" in the sense that the cluster label of an object in the middle of a dense point cloud can assume a cluster label different from those of its neighbors. A related argument has been used to motivate "space-level" constraints in preference to "instance-level" constraints in [148]: the influence of instance-level constraints may fail to propagate to the surrounding points. This problem may also be attributed to the problem of the inconsistent hypothesis space discussed in Section 5.0.2.1, because the cluster labels of points under constraints are determined in a way that is different from the points without constraints. When λ^+ increases further, the cost for this counter-intuitive configuration remains the same, because no constraints are violated. Let C denote the cost of this counter-intuitive configuration.

We are now in a position to understand why it is not possible to attain the desirable configuration "LR". By pushing the vertical and horizontal point clouds away from each other, we can arbitrarily increase the cost of the "LR" configuration, while keeping the cost of the "TB" configuration the same. While the cost for the counter-intuitive configuration also increases when the two point clouds are pushed apart, such an increase is very slow because only the distance of one point (as in the term $||\mathbf{y}_i - \boldsymbol{\mu}_1^{\mathrm{TB}}||^2$) is affected. Consequently, the cost of "LR" configuration can be made larger than C, which is indeed the case for the example in Figure 5.3. Therefore, assuming that the clustering under constraints algorithm finds the clustering solution that minimizes the cost function, the desired "LR" configuration can never be recovered.

Note that specifying additional constraints (either must-link or must-not-link) on points already participating in the constraints cannot solve the problem, because none of the constraints are violated in the counter-intuitive configuration. This problem remained unnoticed in previous studies, because it is a consequence of a small number of constraints. When there are a large number of data points involved in constraints, the sum of the square error is no longer dominated by data points not involved in constraints. The enforcement of constraints changes the cluster labels, which in turn modifies the cluster centers significantly during the minimization of the sum of error. The counter-intuitive configuration is no longer optimal, and the "LR" configuration will be generated because of its smaller cost. Note that this problem is independent of the probabilistic model chosen to represent each cluster: the same problem can arise if there is no restriction on the covariance matrix, for example.

There are several ways to circumvent this problem. One possibility is to increase the number of constraints so that the constraints involve a large number of data points. However, clustering under constraints is most useful when there are few constraints, because the creation of constraints often requires a significant effort on the part of the user. Instead of soliciting additional constraints from the user, the system should provide the user an option to increase arbitrarily the influence of the existing constraints – something the hidden Markov random field approach fails to do. One may also try to initialize the cluster parameters intelligently [13] so that a desired local minimum (the "LR" configuration in Figure 5.3(c)) is obtained, instead of the global minimum (the counter-intuitive configuration in Figure 5.3(d) or the "TB" configuration in Figure 5.3(b), depending on the value of λ^+). However, this approach is heuristic. Indeed, the discussion above reveals a problem in the objective function itself, and we should specify a more appropriate objective function to reflect what the user really desires. The solution in [161] is to introduce a parameter (in addition to λ^+ and λ^-) that can increase the influence of data points in constraints. However, this approach introduces an additional parameter, and it is also heuristic. An alternative potential function for use in the hidden Markov random field has been proposed in [14] to try to circumvent the problem.

Because the main problem lies in the objective function itself, we propose a principled solution to this problem by specifying an alternative objective function for clustering under constraints.

5.3 Proposed Approach

Our approach begins by requiring the hypothesis space (see Section 5.0.2) used by parametric clustering under constraints to be the same as the hypothesis space used by parametric clustering without constraints. This means that the cluster label of an object should be determined by its feature vector and the cluster parameters according to the MAP rule in Equation (5.3) based on the standard finite mixture model in Equation (5.1). The constraints should play no role in deciding the cluster labels. This contrasts with the hidden Markov random field approaches (see Section 5.2), where both the cluster labels and the cluster parameters can freely vary to minimize the cost function.

The desirable cluster parameters should (i) result in cluster labels that satisfy the constraints, and (ii) explain the data well. These two goals, however, may conflict with each other, and a compromise is made by the use of tradeoff parameters. Formally, we seek the parameter vector θ that maximizes an objective function $\mathcal{J}(\theta;\mathcal{Y},\mathcal{C})$, defined by

$$\mathcal{J}(\theta;\mathcal{Y},\mathcal{C}) = \mathcal{L}(\theta;\mathcal{Y}) + \mathcal{F}(\theta;\mathcal{C}), \tag{5.10}$$

$$\mathcal{F}(\theta;\mathcal{C}) = -\sum_{h=1}^{m^+} \lambda_h^+ f^+(\theta;\mathcal{C}_h^+) - \sum_{h=1}^{m^-} \lambda_h^- f^-(\theta;\mathcal{C}_h^-), \tag{5.11}$$

where $\mathcal{F}(\theta;\mathcal{C})$ denotes how well the clusters specified in θ satisfy the constraints in $\mathcal{C}$. It consists of two terms: $f^+(\theta;\mathcal{C}_h^+)$ and $f^-(\theta;\mathcal{C}_h^-)$. The loss functions $f^+(\theta;\mathcal{C}_h^+)$ and $f^-(\theta;\mathcal{C}_h^-)$ correspond to the violation of the h-th must-link constraint (denoted by $\mathcal{C}_h^+$) and the h-th must-not-link constraint (denoted by $\mathcal{C}_h^-$), respectively. There are altogether m^+ must-link constraints and m^- must-not-link constraints, i.e., $|\mathcal{C}_h^+| = m^+$ and $|\mathcal{C}_h^-| = m^-$. The log-likelihood term $\mathcal{L}(\theta;\mathcal{Y})$, which corresponds to the fit of the data $\mathcal{Y}$ by the model parameter θ, is the same as the log-likelihood of the finite mixture model used in standard parametric clustering (Equation (5.2)). The parameters λ_i^+ and λ_i^- give us flexibility to assign different weights on the constraints. In practice, they are set to a common value λ. The value of λ can either be specified by the user, or it can be estimated by a cross-validation type of procedure. For brevity, sometimes we drop the dependence of $\mathcal{J}$ on θ, $\mathcal{Y}$ and $\mathcal{C}$ and write $\mathcal{J}$ as the objective function.

How can this approach be superior to the HMRF approaches? A counter-intuitive clustering solution such as the one depicted in Figure 5.3(d) is no longer attainable. The cluster boundaries are determined solely by the cluster parameters. So, in the example in Figure 5.3(d), the top-left "big plus" point will assume the cluster label of "x", whereas the bottom-right "big cross" point will assume the cluster label of "+", based on the value of the cluster parameters as shown in the figure. The second benefit is that the effect of the instance-level constraints is propagated to the surrounding points automatically, thereby achieving the effect of the desirable space-level constraints. This is because parametric cluster boundaries divide the data space into different contiguous regions. Another advantage of the proposed approach is that it can obtain clustering solutions unattainable by HMRF approaches. For example, the "TB" configuration in Figure 5.3(b) can be made to have an arbitrarily high cost by increasing the value of the constraint penalty parameter λ^+. Since the cost of the "LR" configuration is not affected by λ^+, the "LR" configuration will have a smaller cost than the "TB" configuration with a large λ^+. When the cost function is minimized, the "LR" configuration sought by the user will be returned.

5.3.1 Loss Function for Constraint Violation

What should be the form of the loss functions $f^+(\theta; \mathcal{C}_h^+)$ and $f^-(\theta; \mathcal{C}_h^+)$? Suppose the points $\mathbf{y}_i$ and $\mathbf{y}_j$ participate in a must-link constraint. This must-link constraint is violated if the cluster labels z_i (for $\mathbf{y}_i$) and z_j (for $\mathbf{y}_j$), determined by the MAP rule, are different. Define $\mathbf{z}_i$ to be a vector of length k, such that its l-th entry is one if $z_i = l$, and zero otherwise. The number of constraint violations can be represented by $d(\mathbf{z}_i, \mathbf{z}_j)$ if d is a distance measure such that $d(\mathbf{z}_i, \mathbf{z}_j) = 1$ if $z_i \neq z_j$ and zero, otherwise. Similarly, the violation of a must-not-link constraint between $\mathbf{y}_{i*}$ and $\mathbf{y}_{j*}$ can be represented by $1 - d(\mathbf{z}_{i*}, \mathbf{z}_{j*})$, where $\mathbf{y}_{i*}$ and $\mathbf{y}_{j*}$ are involved in a must-not-link constraint.

Adopting such a distance function $d(.,.)$ as the loss functions $f^+(.)$ and $f^-(.)$ is, however, not a good idea because $d(\mathbf{z}_i, \mathbf{z}_j)$ is a discontinuous function of θ, due to the presence of $\arg\max$ in Equation (5.3). In order to construct an easier optimization problem, we "soften" $\mathbf{z}_i$ and define a new vector $\mathbf{s}_i$ by

$$s_{il} = \frac{\left(\alpha_l p(\mathbf{y}_i|\theta_l)\right)^\tau}{\sum_{l'} \left(\alpha_{l'} p(\mathbf{y}_i|\theta_{l'})\right)^\tau} = \frac{q_{il}^\tau}{\sum_{l'} q_{il'}^\tau}, \tag{5.12}$$

where $q_{il} = \alpha_l p(\mathbf{y}_i|\theta_l)$, and τ is the smoothness parameter. When τ goes to infinity, $\mathbf{s}_i$ approaches $\mathbf{z}_i$, whereas a small value of τ leads to a smooth loss function, which, in general, has a less severe local optima problem.

Another issue is the choice of the distance function $d(\mathbf{s}_i, \mathbf{s}_j)$. Since $s_{il} \geq 0$ and $\sum_l s_{il} = 1$, s_{il} has a probabilistic interpretation. A divergence is therefore more appropriate than a common distance measure such as the Minkowski distance for comparing $\mathbf{s}_i$ and $\mathbf{s}_j$. We adopt the Jensen-Shannon divergence $D_{JS}(\mathbf{s}_i, \mathbf{s}_j)$ [173] with a uniform class prior as the distance measure:

$$\begin{aligned} D_{JS}(\mathbf{s}_i, \mathbf{s}_j) &= \frac{1}{2}\left(\sum_{l=1}^{k} s_{il} \log \frac{s_{il}}{t_l} + \sum_{l=1}^{k} s_{jl} \log \frac{s_{jl}}{t_l}\right) \\ &= \frac{1}{2}\left(\sum_{l=1}^{k} s_{il} \log s_{il} + \sum_{l=1}^{k} s_{jl} \log s_{jl}\right) - \sum_{l=1}^{k} t_l \log t_l, \\ \text{where} \qquad t_l &= \frac{1}{2}(s_{il} + s_{jl}). \end{aligned} \tag{5.13}$$

There are several desirable properties of Jensen-Shannon divergence. It is symmetric, well-defined for all $\mathbf{s}_i$ and $\mathbf{s}_j$, and its square root can be shown to be a metric [76, 199]. The minimum value of 0 for $D_{JS}(.,.)$ is attained only when $\mathbf{s}_i = \mathbf{s}_j$. It is upper-bounded by a constant ($\log 2$), and this happens only when $\mathbf{s}_i$ and $\mathbf{s}_j$ are farthest apart, i.e., when $s_{il} = 1$ and $s_{jh} = 1$ with $l \neq h$. Because $\frac{1}{\log 2} D_{JS}(\mathbf{z}_i, \mathbf{z}_j) = 1$ if $z_i \neq z_j$ and 0 otherwise, the Jensen-Shannon divergence satisfies (up to a multiplicative constant) the desirable property of a distance measure as described earlier in this section. Note that Kullback-Leibler divergence can become unbounded when $\mathbf{s}_i$ and $\mathbf{s}_j$ have different supports, and thus it is not an appropriate choice.

Jensen-Shannon divergence has an additional appealing property: it can be generalized to measure the difference between more than two distributions. This gives a very natural extension to constraints at the group level [231, 166]. Suppose e objects participate in the h-th group-level must-link constraint. This is denoted by the variables a_{hi} introduced in Section 5.1.2, where $a_{hi} = 1/e$ if $\mathbf{y}_i$ participates in this constraint, and zero otherwise. The Jensen-Shannon divergence for the h-th must-link constraint $D^{+}_{JS}(h)$ is defined as

$$\begin{aligned} D^{+}_{JS}(h) &= \sum_{i=1}^{n} a_{hi} \sum_{l=1}^{k} s_{il} \log \frac{s_{il}}{t^{+}_{hl}} = \sum_{i=1}^{n} a_{hi} \sum_{l=1}^{k} s_{il} \log s_{il} - \sum_{l=1}^{k} t^{+}_{hl} \log t^{+}_{hl}, \\ \text{where} \quad t^{+}_{hl} &= \sum_{i=1}^{n} a_{hi} s_{il}. \end{aligned} \tag{5.14}$$

Similarly, the Jensen-Shannon divergence for the h-th must-not-link constraint $D^{-}_{JS}(h)$ is defined as

$$\begin{aligned} D^{-}_{JS}(h) &= \sum_{i=1}^{n} b_{hi} \sum_{l=1}^{k} s_{il} \log \frac{s_{il}}{t^{-}_{hl}} = \sum_{i=1}^{n} b_{hi} \sum_{l=1}^{k} s_{il} \log s_{il} - \sum_{l=1}^{k} t^{-}_{hl} \log t^{-}_{hl}, \\ \text{where} \quad t^{-}_{hl} &= \sum_{i=1}^{n} b_{hi} s_{il}. \end{aligned} \tag{5.15}$$

Here, b_{hi} denotes the must-not-link constraint as discussed in Section 5.1.2. The proposed objective function in Equation (5.10) can be rewritten as

$$\begin{aligned} \mathcal{J} &= \mathcal{L}(\theta; \mathcal{Y}) + \mathcal{F}(\theta; \mathcal{C}) \\ &= \mathcal{L}^{\text{annealed}}(\theta; \mathcal{Y}, \gamma) - \sum_{h=1}^{m^{+}} \lambda^{+}_{h} D^{+}_{JS}(h) + \sum_{h=1}^{m^{-}} \lambda^{-}_{h} D^{-}_{JS}(h), \end{aligned} \tag{5.16}$$

where the annealed log-likelihood $\mathcal{L}^{\text{annealed}}(\theta; \mathcal{Y}, \gamma)$, defined in Equation (B.2), is a generalization of the log-likelihood intended for deterministic annealing. When $\gamma = 1$, $\mathcal{L}^{\text{annealed}}(\theta; \mathcal{Y}, \gamma)$ equals $\mathcal{L}(\theta; \mathcal{Y})$. Note that both $D^{+}_{JS}(h)$ and $D^{-}_{JS}(h)$ are functions of θ.

5.4 Optimizing the Objective Function

The proposed objective function (Equation (5.16)) is more difficult to optimize than the log-likelihood (Equation (5.2)) used in standard parametric clustering. We cannot derive any efficient convex relaxation for $\mathcal{J}$, meaning that a bound-optimization procedure such as the EM algorithm cannot be applied. We resort to general nonlinear optimization algorithms to optimize the objective function. In Section 5.4.1, we shall present the general idea of these algorithms. After describing some details of the algorithms in Section 5.4.2, we present the specific equations used for a mixture of Gaussians in Section 5.4.3. Note that these algorithms are often presented in the literature as minimization algorithms. Therefore, we minimize $-\mathcal{J}$ rather than maximizing $\mathcal{J}$ in practice.

5.4.1 Unconstrained Optimization Algorithms

Different algorithms have been attempted to optimize the proposed objective function $\mathcal{J}$. They include conjugate gradient, quasi-Newton, preconditioned conjugate gradient, and line-search Newton. Because these algorithms are fairly well-documented in the literature [87, 23], we shall only describe their general ideas here. All of these algorithms are iterative and require an initial parameter vector $\theta^{(0)}$.

5.4.1.1 Nonlinear Conjugate Gradient

The key idea of nonlinear conjugate gradient is to maintain the descent directions $\mathbf{d}^{(t)}$ in different iterations, so that different $\mathbf{d}^{(t)}$ are orthogonal (conjugate) to each other with respect to some approximation of the Hessian matrix. This can prevent the inefficient "zig-zag" behavior encountered in steepest descent, which always uses the negative gradient for descent. Initially, $\mathbf{d}^{(0)}$ equals the negative gradient of the function to be minimized. At iteration t, a line-search is performed along $\mathbf{d}^{(t)}$, i.e., we seek η such that the objective function evaluated at $\theta^{(t)} + \eta\mathbf{d}^{(t)}$ is minimized, where $\theta^{(t)}$ is the current parameter estimate. The parameter is then updated by $\theta^{(t+1)} = \theta^{(t)} + \eta\mathbf{d}^{(t)}$. The next direction of descent $\mathbf{d}^{(t+1)}$ is found by computing a vector that is (approximately) conjugate to previous descent directions. Many different schemes have been proposed for this, and we follow the suggestion given in the tutorial [232] and adopt the Polak-Ribiére method with restarting to update $\mathbf{d}^{(t+1)}$:

$$\begin{aligned}
\zeta^{(t+1)} &= \max\left(\frac{(\mathbf{r}^{(t+1)})^T(\mathbf{r}^{(t+1)} - \mathbf{r}^{(t)})}{(\mathbf{r}^{(t)})^T\mathbf{r}^{(t)}}, 0 \right) \\
\mathbf{d}^{(t+1)} &= \mathbf{r}^{(t+1)} + \zeta^{(t+1)}\mathbf{d}^{(t)}.
\end{aligned}$$

Note that line-search in conjugate gradient should be reasonably accurate, in order to ensure that the search directions $\mathbf{d}^{(t)}$ are indeed approximately conjugate (see the discussion in Chapter 7 in [23]).

The main strength of conjugate gradient is that its memory usage is only linear with respect to the number of variables, thereby making it attractive for large scale problems. Conjugate gradient has also found empirical success in fitting a mixture of Gaussians [222], and is shown to be more efficient than the EM algorithm when the clusters are highly overlapping.

5.4.1.2 Quasi-Newton

Consider the second-order Taylor expansion for a real-valued function $f(\mathbf{x})$, which is

$$f(\theta) \approx f(\theta^{(t)}) + (\theta - \theta^{(t)})^T \mathbf{g}(\theta^{(t)}) + \frac{1}{2}(\theta - \theta^{(t)})^T \mathbf{H}(\theta^{(t)})(\theta - \theta^{(t)}), \tag{5.17}$$

where $\mathbf{g}(\mathbf{x})$ and $\mathbf{H}(\mathbf{x}_0)$ denote the gradient and the Hessian of the function $f(.)$ evaluated with $\mathbf{x} = \mathbf{x}_0$. For brevity, we shall drop the reference to $\theta^{(t)}$ for both $\mathbf{g}$ and $\mathbf{H}$. Assuming that $\mathbf{H}$ is positive definite, the right-hand-side of the above approximation can be minimized by $\theta = \theta^{(t)} - \mathbf{H}^{-1}\mathbf{g}$.

The quasi-Newton algorithm does not require explicit knowledge of the Hessian $\mathbf{H}$, which can sometimes be tricky to obtain. Instead, it maintains an approximate Hessian $\widetilde{\mathbf{H}}$, which should satisfy the quasi-Newton condition:

$$\theta^{(t+1)} - \theta^{(t)} = \tilde{\mathbf{H}}^{-1}(\mathbf{g}^{(t+1)} - \mathbf{g}^{(t)}).$$

Since the inversion of the Hessian can be computationally expensive, $\mathbf{G}^{(t)}$, the inverse of the Hessian is approximated instead. While different schemes to update $\mathbf{G}^{(t)}$ are possible, the de facto standard is the BFGS (Broyden-Fletcher-Goldfarb-Shanno) procedure. Below is its description taken from [23]:

$$\begin{aligned}
\mathbf{p} &= \theta^{(t+1)} - \theta^{(t)} \\
\mathbf{v} &= -(\mathbf{g}^{(t+1)} - \mathbf{g}^{(t)}) \\
\mathbf{u} &= \frac{1}{\mathbf{p}^T\mathbf{v}} - \frac{1}{\mathbf{v}^T\mathbf{G}^{(t)}\mathbf{v}}\mathbf{G}^{(t)}\mathbf{v} \\
\mathbf{G}^{(t+1)} &= \mathbf{G}^{(t)} + \frac{1}{\mathbf{p}^T\mathbf{v}}\mathbf{p}\mathbf{p}^T - \frac{1}{\mathbf{v}^T\mathbf{G}^{(t)}\mathbf{v}}\mathbf{G}^{(t)}\mathbf{v}\mathbf{v}^T\mathbf{G}^{(t)} + (\mathbf{v}^T\mathbf{G}^{(t)}\mathbf{v})\mathbf{u}\mathbf{u}^T.
\end{aligned}$$

Given that $\mathbf{G}^{(t)}$ is positive-definite and the round-off error is negligible, the above update guarantees that $\mathbf{G}^{(t+1)}$ is positive-definite. The initial value of the approximated inverse Hessian $\mathbf{G}^{(0)}$ is often set to the identity matrix. Note that an alternative approach to implement quasi-Newton is to maintain the Cholesky decomposition of the approximated Hessian instead. This has the advantage that the approximated Hessian is guaranteed to be positive definite even when the round-off error cannot be ignored.

In practice, the quasi-Newton algorithm is accompanied with a line-search procedure to cope with the error in the Taylor approximation in Equation (5.17) when θ is far away from $\theta^{(t)}$. The descent direction used is $-\tilde{\mathbf{H}}^{-1}\mathbf{g}^{(t)}$. Note that if $\tilde{\mathbf{H}}$ is positive definite, $-\mathbf{g}^{(t)}\tilde{\mathbf{H}}^{-1}\mathbf{g}^{(t)}$ will be always negative and $-\tilde{\mathbf{H}}^{-1}\mathbf{g}^{(t)}$ will be a valid descent direction.

The main drawback of the quasi-Newton method is its memory requirement. The approximate inverse Hessian requires $O(|\theta|^2)$ memory, where $|\theta|$ is the number of variables in θ. This can be slow for high-dimensional θ, which is the case when the data $\mathbf{y}_i$ is of high dimensionality.

5.4.1.3 Preconditioned Conjugate Gradient

Both conjugate gradients and quasi-Newton require only the gradient information of the function to be minimized. Faster convergence is possible if we incorporate the analytic form of the Hessian matrix into the optimization procedure. However, what really can help is not the Hessian, but the inverse of the Hessian. Since the inversion of the Hessian can be slow, it is common to adopt some approximation of the Hessian matrix so that its inversion can be done quickly.

Preconditioned conjugate gradient (PCG) uses an approximation to the inverse Hessian to speed up conjugate gradient. The approximation, also known as the preconditioner, is denoted by $\mathbf{M}$. PCG essentially creates an optimization problem that has $\mathbf{M}^{-1/2}\mathbf{H}\mathbf{M}^{-1/2}$ as the "effective" Hessian matrix and applies conjugate gradient to it, where $\mathbf{H}$ is the Hessian matrix of the original optimization problem. If the "effective" Hessian matrix is close to the identity, conjugate gradient can converge very fast.

We refer the reader to the appendix in [232] for the exact algorithm for PCG. Practical implementation of PCG does not require the computation of $\mathbf{M}^{-1/2}$. Only the multiplication by $\mathbf{M}^{-1}$ is needed. Note that the preconditioner should be positive definite, or the descent direction computed may not decrease the objective function. We can see that there are three requirements for a good conditioner: positive definite, efficient inversion, and good approximation of the Hessian. The first and the third requirements can contradict with each other, because the true Hessian is often not positive-definite unless the objective function is convex. Finding a good preconditioner is an art, and often requires insights into the problem at hand. However, general procedures for creating a preconditioner also exist, which can be based on incomplete Cholesky factorization, for example.

5.4.1.4 Line-search Newton

Line-search Newton is almost the same as the quasi-Newton algorithm, except that the Hessian is provided by the user instead of being approximated by the gradients. There is, however, a catch here. The true Hessian may not be positive-definite, meaning that the minimization problem on the right-hand-side of Equation (5.17) does not have a solution. Therefore, it is common to replace the true Hessian with some approximated version that is positive-definite. Since $\mathbf{H}^{-1}\mathbf{g}$ is to be computed, such an approximation should admit efficient inversion, or at least multiplication by its inverse should be fast. There are two possible ways to obtain such an approximation. We can either add $\xi\mathbf{I}$ to the true Hessian, where ξ is some positive number determined empirically, or we can "repair" $\mathbf{H}$ by adding some terms to it to convert it to a positive-definite matrix.

Note that for both line-search Newton and PCG, the approximated inverse of the Hessian, which takes $O(|\theta|^2)$ memory, need not be formed explicitly. The only thing needed is the ability to be multiplied by the approximated inverse.

5.4.2 Algorithm Details

There are several issues that are common to all these optimization algorithms.

5.4.2.1 Constraints on the Parameters

The algorithms described in Section 5.4.1 are all unconstrained optimization algorithms, meaning that there are no restrictions on the values of θ. However, our optimization problem contains the constraint that the mixture weights α_j are positive and sum to one, and the fact that the precision matrix $\mathbf{\Upsilon}_j$ is symmetric and positive definite. For $\{\alpha_j\}$, we re-parameterize by introducing a set of variables $\{\beta_j\}$ and set

$$\alpha_j = \frac{\exp(\beta_j)}{\sum_l \exp(\beta_l)}. \tag{5.18}$$

For $\boldsymbol{\Upsilon}_j$, we can either re-parameterize by introducing $\mathbf{F}_j$ such that $\boldsymbol{\Upsilon}_j = \mathbf{F}_j\mathbf{F}_j^T$, or we can modify our optimization algorithm to cope with the constraints. The positive-definite constraint is enforced by modifying the line-search routine so that the parameters are always feasible. This is a feasible approach because the precision matrices in a reasonable clustering solution should not become near singular. For the symmetric constraint, it is enforced by requiring that the descent direction in line-search always has symmetric precision matrices.

5.4.2.2 Common Precision Matrix

A common practice of fitting a mixture of Gaussian is to assume a common precision matrix, i.e., the precision matrices of all the k Gaussian components are restricted to be the same, i.e., $\boldsymbol{\Upsilon}_1 = \cdots = \boldsymbol{\Upsilon}_k = \boldsymbol{\Upsilon}$. Instead of the gradient with respect to different $\boldsymbol{\Upsilon}_j$, we need the gradient of $\mathcal{J}$ with respect to $\boldsymbol{\Upsilon}$. This can be done easily because

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Upsilon}} = \sum_{j=1}^{k} \frac{\partial \mathcal{J}}{\partial \boldsymbol{\Upsilon}_j}\frac{\partial \boldsymbol{\Upsilon}_j}{\partial \boldsymbol{\Upsilon}} = \sum_{j=1}^{k} \frac{\partial \mathcal{J}}{\partial \boldsymbol{\Upsilon}_j}.$$

Consequently, Equation (B.12) should be modified to

$$\frac{\partial}{\partial \boldsymbol{\Upsilon}}\mathcal{J} = -\frac{1}{2}\sum_{ij} c_{ij}\mathbf{y}_i\mathbf{y}_i^T + \sum_j \left(\frac{1}{2}\boldsymbol{\mu}_j\boldsymbol{\mu}_j^T + \frac{1}{2}\boldsymbol{\Upsilon}^{-1}\right)\sum_i c_{ij} \tag{5.19}$$

whereas Equation (B.20) should be modified to

$$\frac{\partial}{\partial \boldsymbol{\Upsilon}}\mathcal{J} = \frac{1}{2}\boldsymbol{\Upsilon}^{-1}\sum_{ij} c_{ij} - \frac{1}{2}\sum_{ij} c_{ij}(\mathbf{y}_i - \boldsymbol{\mu}_j)(\mathbf{y}_i - \boldsymbol{\mu}_j)^T. \tag{5.20}$$

The case for Cholesky parameterization is similar. We set $\mathbf{F}_1 = \cdots = \mathbf{F}_k = \mathbf{F}$, and Equation (B.15) should be modified to

$$\frac{\partial}{\partial \mathbf{F}}\mathcal{J} = -\sum_{ij} c_{ij}\mathbf{y}_i\mathbf{y}_i^T\mathbf{F} + \sum_j \left(\boldsymbol{\mu}_j\boldsymbol{\mu}_j^T + \boldsymbol{\Upsilon}^{-1}\right)\mathbf{F}\sum_i c_{ij}, \tag{5.21}$$

and Equation (B.21) should be modified to

$$\frac{\partial}{\partial \mathbf{F}}\mathcal{J} = \boldsymbol{\Upsilon}^{-1}\mathbf{F}\sum_{ij} c_{ij} - \sum_{ij} c_{ij}(\mathbf{y}_i - \boldsymbol{\mu}_j)(\mathbf{y}_i - \boldsymbol{\mu}_j)^T\mathbf{F}. \tag{5.22}$$

5.4.2.3 Line Search Algorithm

The line-search algorithm we used is based on the implementation in Matlab, which is in turn based on section 2.6 in [86]. Its basic idea is to perform a cubic interpolation based on the value of the function and the gradient evaluated at two parameter values. The line search terminates when the Wolfe's condition is satisfied. Following the advice in Chapter 7 of [23], the line-search is stricter for both conjugate gradient and preconditioned conjugate gradient in order to ensure conjugacy. Note that when the Gaussians are parameterized by their precision matrices, the line search procedure disallows any parameter vector that has non-positive definite precision matrices.

5.4.2.4 Annealing the Objective Function

The algorithms described in Section 5.4.1 find only the local minima of $\mathcal{J}$ based on the initial parameter estimate $\theta^{(0)}$. One strategy to alleviate this problem is to adopt a deterministic-annealing type of procedure and use a “smoother” version of the objective function. The solution of this “smoother” optimization problem is used as the initial guess for the actual objective function to be optimized. Specifically, we adjust the two temperature-like parameters γ and τ in $\mathcal{J}$ defined in Equation (5.16). When γ and τ are small, $\mathcal{J}$ is smooth and is almost convex, therefore it is easy to optimize. The annealing stops when γ reaches one and τ reaches a pre-specified value τ^{final}, which is set to four in our experiment. This is, however, a fairly insensitive parameter. Any number between one and sixteen leads to similar clustering results.

5.4.3 Specifics for a Mixture of Gaussians

All the algorithms described in Section 5.4.1 require the gradient information of the objective function. In Appendix B.1, we have derived the gradient information with the assumption that each mixture component is a Gaussian distribution. Recall that $q_{ij} = \log p(\mathbf{y}_i|\theta_j)$, and s_{ij} has been defined in Equation (5.12). Define the following:

$$\begin{aligned}
\tilde{r}_{ij} &= \frac{q_{ij}^{\gamma}}{\sum_{j'} q_{ij'}^{\gamma}} \\
w_{ij} &= \left(\sum_{h=1}^{m^+} \lambda_h^+ a_{hi} - \sum_{h=1}^{m^-} \lambda_h^- b_{hi}\right) s_{ij} \log s_{ij} \\
&\quad - s_{ij} \left(\sum_{h=1}^{m^+} \lambda_h^+ a_{hi} \log t_{hj}^+ - \sum_{h=1}^{m^-} \lambda_h^- b_{hi} \log t_{hj}^-\right) \\
c_{ij} &= \tilde{r}_{ij} - \tau\Big(w_{ij} - s_{ij} \sum_{l=1}^{k} w_{il}\Big).
\end{aligned}$$

The partial derivative of $\mathcal{J}$ with respect to β_j is

$$\frac{\partial \mathcal{J}}{\partial \beta_l} = \sum_i c_{il} - \alpha_l \sum_{ij} c_{ij}. \tag{5.23}$$

Under the natural parameterization $\boldsymbol{\nu}_l$ and $\boldsymbol{\Upsilon}_l$ for the parameters of the l-th cluster, we have

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\nu}_l} = \sum_i c_{il}\mathbf{y}_i - \boldsymbol{\mu}_l \sum_i c_{il} \tag{5.24}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Upsilon}_l} = -\frac{1}{2}\sum_i c_{il}\mathbf{y}_i\mathbf{y}_i^T + \frac{1}{2}\left(\boldsymbol{\mu}_l\boldsymbol{\mu}_l^T + \boldsymbol{\Sigma}_l\right)\sum_i c_{il}. \tag{5.25}$$

If the Cholesky parameterization $\mathbf{F}_l$ is used instead of $\boldsymbol{\Upsilon}_l$, we have

$$\frac{\partial \mathcal{J}}{\partial \mathbf{F}_l} = -\sum_i c_{il}\mathbf{y}_i\mathbf{y}_i^T\mathbf{F}_l + \left(\boldsymbol{\mu}_l\boldsymbol{\mu}_l^T + \boldsymbol{\Sigma}_l\right)\mathbf{F}_l \sum_i c_{il}. \tag{5.26}$$

If the moment parameterization $\boldsymbol{\mu}_l$ and $\boldsymbol{\Upsilon}_l$ are used instead, we have

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\mu}_l} = \boldsymbol{\Upsilon}_l \sum_i c_{il}(\mathbf{y}_i - \boldsymbol{\mu}_l) \tag{5.27}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Upsilon}_l} = \frac{1}{2}\boldsymbol{\Sigma}_l \sum_i c_{il} - \frac{1}{2}\sum_i c_{il}(\mathbf{y}_i - \boldsymbol{\mu}_l)^T(\mathbf{y}_i - \boldsymbol{\mu}_l) \tag{5.28}$$

and the corresponding partial derivative if Cholesky parameterization is used is

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Upsilon}_l} = \left(\boldsymbol{\Sigma}_l \sum_i c_{il} - \sum_i c_{il}(\mathbf{y}_i - \boldsymbol{\mu}_l)^T(\mathbf{y}_i - \boldsymbol{\mu}_l)\right)\mathbf{F}_l. \tag{5.29}$$

The Hessian of $\mathcal{J}$ is clumsier to present, and the reader can refer to Appendix B.2 for its exact form under various parameterizations.

5.5 Feature Extraction and Clustering with Constraints

It turns out that the objective function introduced in Section 5.3 can be modified to simultaneously perform feature extraction and clustering with constraints. There are three reasons why we are interested in performing these two tasks together.

First, the proposed algorithm does not perform well for small data sets with a large number of features (denoted by d), because the d by d covariance matrix is estimated from the available data. In other words, we are suffering from the curse of dimensionality. The standard solution is to preprocess the data by reducing the dimensionality using methods like principal component analysis. However, the resulting low-dimensional representation may not be optimal for clustering with the given set of constraints. It is desirable to incorporate the constraints in seeking a good low-dimensional representation.

The second reason is from a modeling perspective. One can argue that it is inappropriate to model the two desired clusters shown in Figure 5.3(d) by two Gaussians, because the distribution of the data points are very "non-Gaussian": there are no data points in the central regions of the two Gaussians, which are supposed to have the highest data densities! If the data points are projected to the one-dimensional subspace of the x-axis, the resulting two clusters follow the Gaussian assumption well while satisfying the constraints. Note that PCA selects a projection direction that is predominantly based on the y-axis because the data variance in that direction is large. However, the clusters formed after such a projection will violate the constraints. In general, it is quite possible that given a high dimensional data set, there exists a low-dimensional subspace such that the clusters after projection are Gaussian-like, and the constraints are satisfied by those clusters.

The third reason is that the projection can be combined with the kernel trick to achieve clusters with arbitrary shapes. A nonlinear transformation is applied to the data set to embed the data in a high-dimensional feature space. A linear subspace of the given feature space is sought such that the Gaussian clusters formed in that subspace are consistent with the given set of constraints. Because of the non-linear transformation, linear cluster boundaries in that subspace correspond to nonlinear boundaries in the original input space. The exact form of the nonlinear boundaries is controlled by the type of the nonlinear transformation applied. Note that such transformation need not be performed explicitly because of the kernel trick (see Section 2.5.1). In practice, kernel PCA is first performed on the data in order to extract the main structure in the high dimensional feature space. The number of features returned by kernel PCA should be large. The feature extraction algorithm in this section is then applied to the result of kernel PCA.

5.5.1 The Algorithm

Let $\mathbf{x}_i$ be the result of projecting the data point $\mathbf{y}_i$ into a d'-dimensional space, where d' is small and $d' < d$, and d is the dimension of $\mathbf{y}_i$. Let $\mathbf{P}^T$ be the d' by d projection matrix, i.e., $\mathbf{x}_i = \mathbf{P}^T \mathbf{y}_i$, and $\mathbf{P}^T\mathbf{P} = \mathbf{I}$. Let $\mathbf{P}^T\boldsymbol{\mu}_j$ and $\boldsymbol{\Upsilon}$ be the cluster center of the j-th Gaussian component and the common covariance matrix, respectively. Let $\mathbf{R}$ be the Cholesky decomposition of $\boldsymbol{\Upsilon}$, i.e., $\boldsymbol{\Upsilon} = \mathbf{R}\mathbf{R}^T$. We have

$$p(\mathbf{x}_i|z_i = j) = (2\pi)^{-d'/2}(\det \boldsymbol{\Upsilon})^{1/2} \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{P}^T\boldsymbol{\mu}_j)^T \boldsymbol{\Upsilon}(\mathbf{x}_i - \mathbf{P}^T\boldsymbol{\mu}_j)\right). \tag{5.30}$$

Because $\boldsymbol{\Upsilon} = \mathbf{R}^T\mathbf{P}^T\mathbf{P}\mathbf{R}$, we can rewrite the above as

$$\begin{aligned} &\log p(\mathbf{x}_i|z_i = j) \\ &= -\frac{d'}{2}\log(2\pi) + \frac{1}{2}\log\det\boldsymbol{\Upsilon} - \frac{1}{2}(\mathbf{y}_i - \boldsymbol{\mu}_j)^T\mathbf{P}\boldsymbol{\Upsilon}\mathbf{P}^T(\mathbf{y}_i - \boldsymbol{\mu}_j), \\ &= -\frac{d'}{2}\log(2\pi) + \frac{1}{2}\log\det\mathbf{F}^T\mathbf{F} - \frac{1}{2}(\mathbf{y}_i - \boldsymbol{\mu}_j)^T\mathbf{F}\mathbf{F}^T(\mathbf{y}_i - \boldsymbol{\mu}_j), \end{aligned} \tag{5.31}$$

where $\mathbf{F} = \mathbf{P}\mathbf{R}$. Note the similarity between this expression and that of $\log p(\mathbf{y}_i|z_i = j)$ if we adopt the parameterization $\boldsymbol{\Upsilon} = \mathbf{F}\mathbf{F}^T$ as discussed in Section 5.4.2.1. We have

$$\frac{\partial}{\partial\boldsymbol{\mu}_l}\log p(\mathbf{x}_i|z_i = j) = \mathbf{F}\mathbf{F}^T(\mathbf{y}_i - \boldsymbol{\mu}_j), \tag{5.32}$$

$$\frac{\partial}{\partial\mathbf{F}}\log p(\mathbf{x}_i|z_i = j) = \mathbf{F}(\mathbf{F}^T\mathbf{F})^{-1} - (\mathbf{y}_i - \boldsymbol{\mu}_j)(\mathbf{y}_i - \boldsymbol{\mu}_j)^T\mathbf{F}. \tag{5.33}$$

While $\mathbf{P}$ has an orthogonality constraint, there is no constraint on $\mathbf{F}$, and thus we cast our optimization problem in terms of $\mathbf{F}$. The parameters $\mathbf{F}$, $\boldsymbol{\mu}_j$ and β_j can be found by optimizing $\mathcal{J}$, after substituting Equation (5.31) as $\log q_{ij}$ into Equation (5.16). In practice, the quasi-Newton algorithm is used to find the parameters that minimize the objective function, because it is difficult to inverse the Hessian efficiently.

It is interesting to point out that this subspace learning procedure is related to linear discriminant analysis if the data points $\mathbf{y}_i$ are standardized to have equal variance. If we fix $\boldsymbol{\Upsilon}$ to be the identity matrix, maximizing the log-likelihood is the same as minimizing $(\mathbf{y}_i - \boldsymbol{\mu}_j)^T\mathbf{P}^T\mathbf{P}(\mathbf{y}_i - \boldsymbol{\mu}_j)$. This is the within-class scatter of the j-th cluster. Since the sum of between-class scatter and the within-class scatter is the total data scatter, which is constant because of the standardization, maximization of the within-class scatter is the same as maximizing the ratio of between-class scatter to the within-class scatter. This is what linear discriminant analysis does.

5.6 Experiments

To verify the effectiveness of the proposed approach, we have applied our algorithm to both synthetic and real world data sets. We compare the proposed algorithm with two state-of-the-art algorithms for clustering under constraints. The first one, denoted by `Shental`, is the algorithm proposed by Shental *et al.* in [231]. It uses "chunklets" to represent the cluster labels of the objects involved in must-link constraints, and a Markov network to represent the cluster labels of objects participating in must-not-link constraints. The EM algorithm is used for parameter estimation, and the E-step is done by computations within the Markov network. It is not clear from the paper the precise algorithm used for the inference in the E-step, though the Matlab implementation[3] provided by the authors seems to use the junction tree algorithm. This can take exponential time when the constraints are highly coupled. This potential high time complexity is the motivation for the mean-field approximation used in the E-step of [161]. The second algorithm, denoted by `Basu`, is the constrained k-means algorithm with metric learning[4] described in [21]. It is based on the idea of hidden Markov random field, and it uses the constraints to adjust the metrics between different data points. A parameter is needed for the strength of the constraints. Note that we do not compare our approach with the algorithm in [161], because its implementation is no longer available.

5.6.1 Experimental Result on Synthetic Data

Our first experiment is based on the example in Figure 5.3(a), which contains 400 points generated by four Gaussians centered at $\begin{bmatrix} 2 \\ 8 \end{bmatrix}$, $\begin{bmatrix} 2 \\ -8 \end{bmatrix}$, $\begin{bmatrix} -2 \\ -8 \end{bmatrix}$ and $\begin{bmatrix} -2 \\ 8 \end{bmatrix}$, each with identity covariance matrix. Recall that the goal is to group this data set into two clusters – a "left" and a "right" cluster – based on the two must-link constraints. Specifically, points with negative and positive horizontal co-ordinates are intended to be in two different clusters. Note that this synthetic example differs from the similar one in [161] in that the vertical separation between the top and bottom point clouds is larger. This increases the difference between the goodness of the "left/right" and "top/bottom" clustering solutions, so that a small number of constraints is no longer powerful enough to bias one clustering solution over the other as in [161]. The results of running the algorithms `Shental` and `Basu` are shown in Figures 5.4(a) and 5.4(b), respectively. For `Shental` the two Gaussians estimated are also shown. Not only did both algorithms fail to recover the desired cluster structure, but also the cluster assignments found were counter-intuitive. This failure is due to the fact that these two approaches represent the constraints by imposing prior distributions on the cluster labels, as explained earlier in Section 5.2.

The result of applying the proposed algorithm to this data set with $\lambda = 250$ is shown in Figure 5.4(c). The two desired clusters have been almost perfectly recovered, when we compare the solution visually with the desired cluster structure in Figure 5.3(c). A more careful comparison is done in Figure 5.4(d), where the cluster boundaries obtained by the proposed algorithm (the gray dotted line) is compared with the ground-truth (the solid green line). We can see that these two boundaries are very close to each other, indicating that the proposed algorithm discovered a good cluster boundary. This compares with the similar example in [167], where the cluster boundary there (as inferred from the Gaussians shown) is quite different[5] from the desired cluster boundary. An additional cluster boundary obtained by the proposed algorithm when τ took the intermediate value of 1 is also shown (the magenta dashed line). (The final cluster boundary was produced with $\tau = 4$.) This boundary is significantly different from the ground-truth boundary. So, a large value of τ improves the clustering result in this case. This improvement is the consequence of the fact that a large τ focuses on the cluster assignments of the objects and reduces the spurious influence of the exact locations of the points. The Jensen-Shannon divergence measures the constraint violation/satisfaction more accurately. Note that a larger value of τ does not have any further visible effect on the cluster boundary.

[3]The url is `http://www.cs.huji.ac.il/~tomboy/code/ConstrainedEM_plusBNT.zip`.

[4]Its implementation is available at `http://www.cs.utexas.edu/users/ml/risc/code/`.

[5]Note that the synthetic data example in [167] is fitted with a mixture model with different covariance matrices per class. Therefore, comparing it with the proposed algorithm may not be the most fair.

The Gaussian distributions contributing to these cluster boundaries are shown in Figure 5.4(e). We observe that the Gaussians recovered by the proposed algorithm (dotted gray lines) are slightly "fatter" than those obtained with the ground-truth labels (solid green lines). This is because data points not in a particular cluster can still contribute, though to a smaller extent, to the covariance of the Gaussian distributions due to the soft-assignment implied in the mixture model. This is not the case when the covariance matrix is estimated based on the ground-truth labels.

While the proposed algorithm is the only clustering under constraints algorithm we know that can return the two desired clusters, we want to note that a sufficiently large λ is needed for its success. If $\lambda = 50$, for example, the result of the proposed algorithm is shown in Figure 5.4(f). This is virtually identical to the clustering solution without any constraints (Figure 5.3(b)). While the constraints are violated, the clustering solution is more "reasonable" than the solutions shown in Figures 5.4(a) and 5.4(b). Note that it is easy to detect that λ is too small in this example, because the constraints are violated. We should increase λ until this is no longer the case. The resulting clustering solution will effectively be identical to the desired solution shown in Figure 5.4(c).

5.6.2 Experimental Results on Real World Data

We have also compared the proposed algorithm with the algorithms `Shental` and `Basu` based on real world data sets obtained from different domains. The label information in these data sets is used only for the creation of the constraints and for performance evaluation. In particular, the labels are not used by the clustering algorithms.

5.6.2.1 Data Sets Used

Table 5.2 summarizes the characteristics of the data sets used. The following preprocessing has been applied to the data whenever necessary. If a data set has a nominal feature that can assume c possible values with $c > 2$, that feature is converted into c continuous features. The i-th such feature is set to one when the nominal feature assumes the i-th possible value, and the remaining $c - 1$ continuous features are set to zero. If the variances of the features of a data set are very different, standardization is applied to all the features, so that the variances or the ranges of the preprocessed features become the same. If the number of features is too large when compared with the number of data points n, principal component analysis (PCA) is applied to reduce the dimensionality. The number of reduced dimension d is determined by finding the largest d that satisfies $n > 3d^2$, while the principal components with negligible eigenvalues are also discarded. The difficulty of the classification tasks associated with these data sets can be seen by the values of the F-score and the normalized mutual information (to be defined in Section 5.6.2.3) computed using the ground-truth labels, under the assumption that the class conditional densities are Gaussian with common covariance matrices.

Data Sets from UCI The following data sets are obtained from the UCI machine learning repository[6]. The list below includes most of the data sets in the repository that have mostly continuous features and have relatively balanced class sizes.

The dermatology database (`derm`) contains 366 cases with 34 features. The goal is to determine the type of Erythemato-Squamous disease based on the features extracted. The age attribute, which has missing values, is removed. PCA is performed to reduce the resulting 33 dimensional data to 11 features. The sizes of the six classes are 112, 61, 72, 49, 52 and 20.

[6]The url is `http://www.ics.uci.edu/~mlearn/MLRepository.html`

The optical recognition of handwritten digits data set (`digits`) is based on normalized bitmaps of handwritten digits extracted from a preprinted form. The 32x32 bitmaps are divided into non-overlapping blocks of 4x4 and the number of pixels are counted in each block. Thus 64 features are obtained for each digit. The training and testing sets are combined, leading to 5620 patterns. PCA is applied to reduce the dimensionality to 42 to preserve 99% of the total variance. The sizes of the ten classes are 554, 571, 557, 572, 568, 558, 558, 566, 554, and 562.

The ionosphere data set (`ion`) consists of 351 radar readings returned from the ionosphere. seventeen pulse numbers are extracted from each reading. The real part and the imaginary part of the complex pulse numbers constitute the 34 features per pattern. There are two classes: "good" radar returns (225 patterns) are those showing evidence of some type of structure in the ionosphere, and "bad" returns (126 patterns) are those that do not; their signals pass through the ionosphere. PCA is applied to reduce the dimensionality to 10.

The multi-feature digit data set consists of features of handwritten numerals extracted from a collection of Dutch utility maps. Multiple types of features have been extracted. We have only used the features based on the 76 Fourier coefficients of the character shapes. The resulting data set is denoted by `mfeat-fou`. There are 200 patterns per digit class. PCA is applied to reduce the dimensionality to 16, which preserves 95% of the total energy.

The Wisconsin breast cancer diagnostic data set (`wdbc`) has two classes: benign (357 cases) and malignant (212 cases). The 30 features are computed from a digitized image of the breast tissue, which describes the characteristics of the cell nuclei present in the image. All the features are standardized to have mean zero and variance one. PCA is applied to reduce the dimensionality of the data to 14.

The UCI image segmentation data set (`UCI-seg`) contains 19 continuous attributes extracted from random 3x3 regions of seven outdoor images. One of the features has zero variation and is discarded. The training and testing sets are combined to form a data set with 2310 patterns. After standardizing each feature to have variance one, PCA is applied to reduce the dimensionality of the data to 10. The seven classes correspond to brick-face, sky, foliage, cement, window, path, and grass. Each of the classes has 330 patterns.

Data Sets from Statlog in UCI The following five data sets are taken from the Statlog section[7] in the UCI machine learning repository.

The Australian credit approval data set (`austra`) has 690 instances with 14 attributes. The two classes are of size 383 and 307. The continuous features are standardized to have standard deviation 0.5. Four of the features are non-binary nominal features, and they are converted to multiple continuous features. PCA is then applied to reduce the dimensionality of the concatenated feature vector to 15.

The German credit data (`german`) contains 1000 records with 24 features. The version with numerical attributes is used in our experiments. PCA is used to reduce the dimensionality of the data to 18, after standardizing the features so that all of them lie between zero and one. The two classes have 700 and 300 records.

The heart data set (`heart`) has 270 observations with 13 raw features in two classes with 150 and 120 data points. The three nominal features are converted into continuous features. The continuous features are standardized to have standard deviation 0.5, before applying PCA to reduce the data set to 9 features.

[7]The url is `http://www.ics.uci.edu/~mlearn/databases/statlog/`

The satellite image data set (`sat`) consists of the multi-spectral values of pixels in 3x3 neighborhoods in a satellite image. The aim is to classify the class associated with the central pixel, which can be "red soil", "cotton crop", "grey soil", "damp grey soil", "soil with vegetation stubble" or "very damp grey soil". The training and the testing sets are combined to yield a data set of size 6435. There are 36 features altogether. The classes are of size 1533, 703, 1358, 626, 707 and 1508.

The vehicle silhouettes data set (`vehicle`) contains a set of features extracted from the silhouette of a vehicle. The goal is to classify a vehicle as one of the four types (Opel, Saab, bus, or van) based on the silhouette. There are altogether 846 patterns in the four classes with sizes of the four classes as 212, 217, 218, and 199. The features are first standardized to have standard deviation one, before applying PCA to reduce the dimensionality to 16.

Other Data Sets We have also experimented the proposed algorithm with data sets from other sources.

The texture classification data set (`texture`) is taken from [127]. It consists of 4000 patterns with four different types of textures. The 19 features are based on Gabor filter responses. The four classes are of sizes 987, 999, 1027, and 987.

The online handwritten script data set (`script`), taken from [192], is about a problem that classifies words and lines in an online handwritten document into one of the six major scripts: Arabic, Cyrillic, Devnagari, Han, Hebrew, and Roman. Eleven spatial and temporal features are extracted from the strokes of the words. There are altogether 12938 patterns, and the sizes of the six classes are 1190, 3173, 1773, 3539, 1002, and 2261.

The ethnicity recognition data set (`ethn`) was originally used in [175]. The goal is to classify a 64x64 face image into two classes: "Asian" (1320 images) and "non-Asian" (1310 images). It includes the PF01 database[8], the Yale database[9], the AR database [181], and the non-public NLPR database[10]. Some example images are shown in Figure 5.5. 30 eigenface coefficients are extracted to represent the images.

The clustering under constraints algorithm is also tested on an image segmentation task based on the Mondrian image shown in Figure 5.6, which has five distinct segments. The image is divided into 101 by 101 sites. Twelve histogram features and twelve Gabor filter responses of four orientations at three different scales are extracted. Because the histogram features always sum to one, PCA is performed to reduce the dimension from 24 to 23. The resulting data set `Mondrian` contains 10201 patterns with 23 features in 5 classes. The sizes of the classes are 2181, 2284, 2145, 2323, and 1268.

The 3-newsgroup database[11] is about the classification of Usenet articles from different newsgroups. It has been used previously to demonstrate the effectiveness of clustering under constraints in [14]. It consists of three classification tasks (`diff-300`, `sim-300`, `same-300`), each of which contains roughly 300 documents from three different topics. The topics are regarded as the classes to be discovered. The three classification tasks are of different difficulties: the sets of three topics in `diff-300`, `sim-300`, and `same-300` respectively have increasing similarities. Latent semantic indexing is applied to the tf-idf normalized word features to convert each newsgroup article into a feature vector of dimension 10. The three classes in `diff-300` are all of sizes 100, whereas the number of patterns in the three classes in `sim-300` is 96, 97, and 98. The sizes of the classes in `same-300` are 99, 98, and 100.

[8]`http://nova.postech.ac.kr/archives/imdb.html`.

[9]`http://cvc.yale.edu/projects/yalefaces/yalefaces.html`.

[10]Provided by Dr. Yunhong Wang, National Laboratory for Pattern Recognition, Beijing.

[11]It can be downloaded at `http://www.cs.utexas.edu/users/ml/risc/`.

Notice that the data sets `ethn`, `Mondrian`, `diff-300`, `sim-300`, and `same-300` have been used in the previous work [161]. The same preprocessing is applied for both `ethn` and `Mondrian`as in [161], though we reduce the dimensionality of the data set from 20 to 10 for the `diff-300`, `sim-300`, and `same-300` data sets based on our "$n > 3d^2$" rule.

5.6.2.2 Experimental Procedure

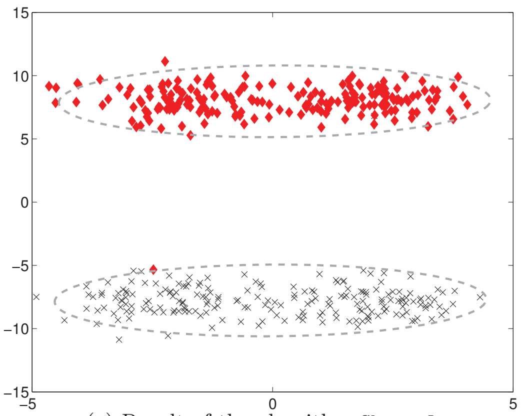
For each data set listed in Table 5.2, a constraint was specified by first generating a random point pair $(\mathbf{y}_i, \mathbf{y}_j)$. If the ground-truth class labels of $\mathbf{y}_i$ and $\mathbf{y}_j$ were the same, a must-link constraint was created between $\mathbf{y}_i$ and $\mathbf{y}_j$. Otherwise, a must-not-link constraint was created. Different numbers of constraints were created as a percentage of the number of points in the data set: 1%, 2%, 3%, 5%, 10%, and 15%. Note that the constraints were generated in a "cumulative" manner: the set of "1%" constraints was included in the set of "2%" constraints, and so on.

The line-search Newton algorithm was used to optimize the objective function $\mathcal{J}$ in the proposed approach. The Gaussians were represented by the natural parameters $\boldsymbol{\nu}_j$ and $\boldsymbol{\Upsilon}$, with a common precision matrix among different Gaussian components. This particular choice of optimization algorithm was made based on a preliminary efficiency study, where this approach was found to be the most efficient among all the algorithms described in Section 5.4.1. Because the gradient is available in line-search Newton, convergence was decided when the norm of the gradient was less than a threshold of the norm of the initial gradient. Note that this is a stricter and more reasonable convergence criteria than the one typically used in the EM algorithm, which is based on the relative change of log-likelihood. However, in order to safeguard against round-off error, we also declare convergence when the relative change of the objective function is very small: 10^{-10}, to be precise. Starting with a random initialization, line-search Newton was run with $\gamma = 1$ and $\tau = 0.25$, with the convergence threshold set to 10^{-2}. Line-search Newton was run again after increasing τ to 1, with the convergence threshold tightened to 10^{-3}. Finally, τ and the convergence threshold were set to 4 and 10^{-4}, respectively. The optimization algorithm was also stopped if convergence was not achieved within 5000 Newton iterations. Fifteen random initializations were attempted. The solution with the best objective function value was regarded as the solution found by the proposed algorithm.

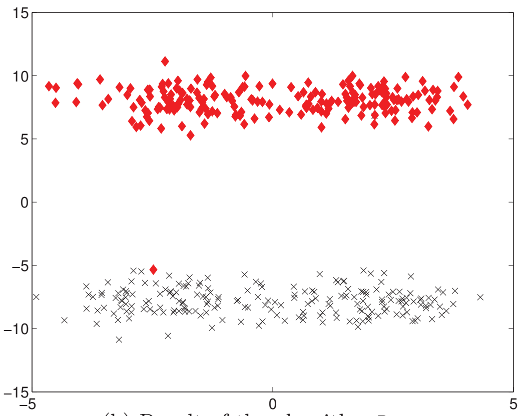
The above procedure, however, assumes the constraint strength λ is known. The value of λ was determined using a set of validation constraints. The constraints for training set and the constraints for validation set were obtained using the following rules. Given a data set, if the number of constraints was less than $3k$, k being the number of clusters, all the available constraints were used for training and validation. This procedure, while risking overfitting, is necessary because a too small set of constraints is poor for training the clusters as well as the estimation of λ. When the number of constraints was between $3k$ and $6k$, the number of training constraints and validation constraints were both set to $3k$. So, the training constraints overlapped with the validation constraints. When the number of constraints was larger than $6k$, half the constraints were used for training and the other half were used for validation. Starting with $\lambda = 0.1$, we increased λ by multiplying it by $\sqrt{10}$. For each λ, the proposed algorithm was executed. A better value of λ was encountered if the number of violations of the validation constraints was smaller than the current best. If there was a tie, the decision was made on the number of violations of the training constraints. If the best value of λ did not change for four iterations, we assumed that the optimal value of λ was found. The proposed algorithm was executed again using all the available constraints and λ value just determined. The resulting solution was compared with the solution obtained using only the training constraints, and the one with the smaller total number of constraint violations was regarded as our final clustering

	Full name	Source	n	d	k	F	NMI
`derm`	dermatology	UCI	366	11	6	0.9648	0.9258
`digits`	optical recognition of handwritten digits	UCI	5620	42	10	0.9516	0.8915
`ion`	ionosphere	UCI	351	10	2	0.8519	0.4003
`mfeat-fou`	multi-feature digit, Fourier coefficients	UCI	2000	16	10	0.7999	0.7369
`UCI-seg`	UCI Image segmentation	UCI	2310	10	7	0.8445	0.7769
`wdbc`	Wisconsin breast cancer diagnostic	UCI	569	14	2	0.9645	0.8047
`austra`	Australian credit approval	Statlog in UCI	690	15	2	0.8613	0.4384
`german`	German credit	Statlog in UCI	1000	18	2	0.7627	0.1475
`heart`	heart	Statlog in UCI	270	9	2	0.8550	0.4010
`sat`	satellite image	Statlog in UCI	6435	36	6	0.8382	0.7176
`vehicle`	vehicle silhouettes	Statlog in UCI	846	16	4	0.7869	0.5850
`script`	online handwritten script	[192]	12938	11	6	0.7673	0.5812
`texture`	texture	[127]	4000	19	4	0.9820	0.9274
`ethn`	ethnicity	[175]	2630	30	2	0.9627	0.7704
`Mondrian`	Mondrian image segmentation	[161]	10201	23	5	0.9696	0.9042
`diff-300`	Usenet newsgroup (highly different)	[14]	300	10	3	0.9432	0.7895
`sim-300`	Usenet newsgroup (somewhat similar)	[14]	291	10	3	0.6996	0.3290
`same-300`	Usenet newsgroup (highly similar)	[14]	297	10	3	0.7825	0.4071

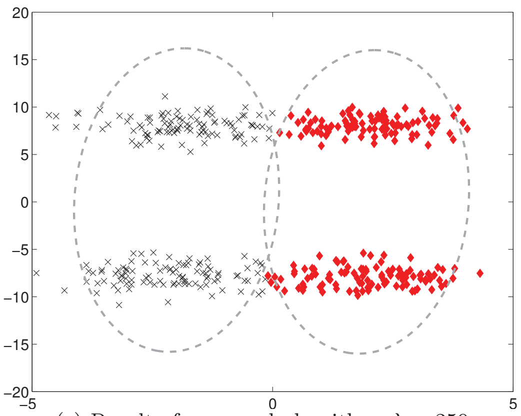
Table 5.2: Summary of the real world data sets used in the experiments. The number of data points and the number of actual features used are represented by n and d, respectively. The difficulty of the classification task associated with a data set can be seen by the F-score (denoted by F) and the normalized mutual information (denoted by NMI). These two criteria are defined in Section 5.6.2.3. Higher values of F and NMI indicate that the associated classification task is relatively easier, $0 < \mathrm{F} \leq 1$ and $0 \leq NMI \leq 1$.



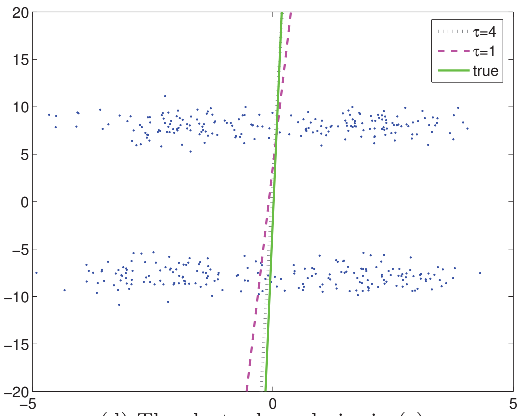
(a) Result of the algorithm `Shental`



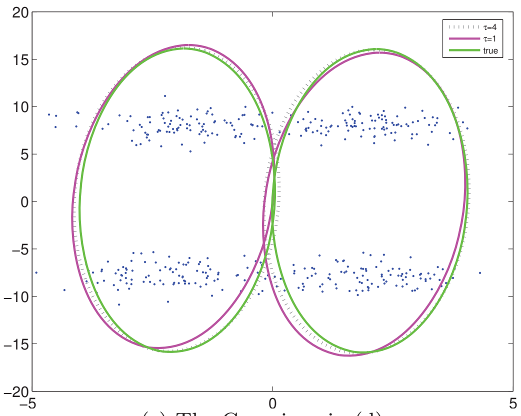
(b) Result of the algorithm `Basu`



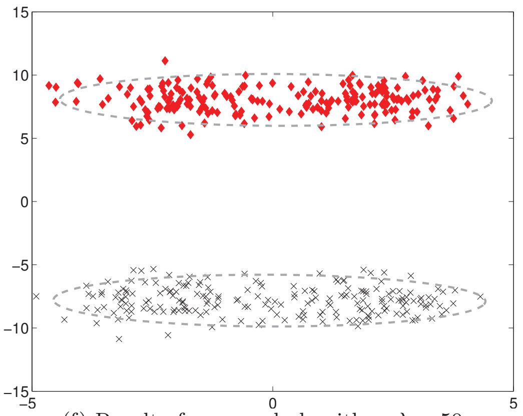
(c) Result of proposed algorithm, $\lambda = 250$



(d) The cluster boundaries in (c)



(e) The Gaussians in (d)



(f) Result of proposed algorithm, $\lambda = 50$

Figure 5.4: The result of running different clustering under constraints algorithms for the synthetic data set shown in Figure 5.3(a). While the algorithms `Shental` and `Basu` failed to discover the desired clusters ((a) and (b)), the proposed algorithm succeeded with $\lambda = 250$ (c). The resulting cluster boundaries and Gaussians are compared with those estimated with the ground-truth labels ((d) and (e)). When $\lambda = 50$, the proposed algorithm returned the natural clustering solution (f).

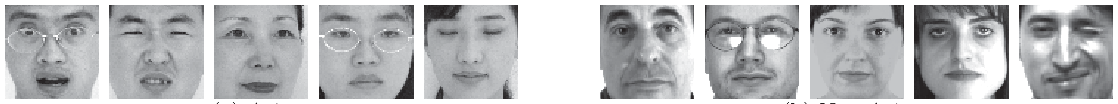


Figure 5.5: Example face images in the ethnicity classification problem for the data set `ethn`.

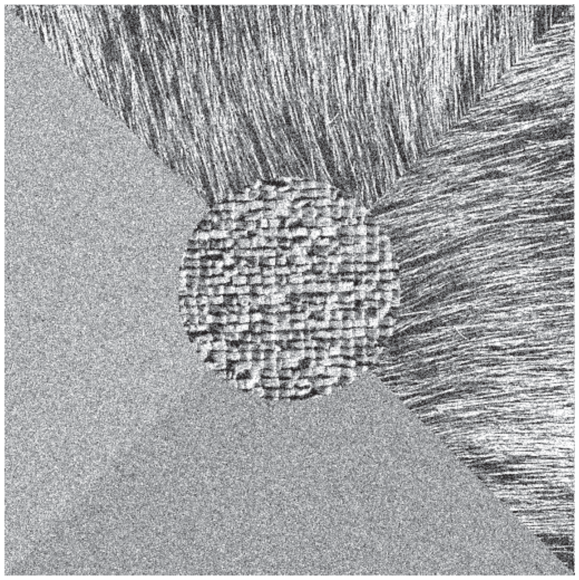


Figure 5.6: The Mondrian image used for the data set `Mondrian`. It contains 5 segments. Three of the segments are best distinguished by Gabor filter responses, whereas the remaining two are best distinguished by their gray-level histograms.

solution. If there was a tie, the solution obtained with training constraints only was selected.

The algorithms `Shental` and `Basu` were run using the same set of data and constraints as input. For `Shental`, we modified the initialization strategy in their software, which involved a two step process. First, five random parameter vectors were generated, and the one with the highest log-likelihood was selected as the initial value of the EM algorithm. Convergence was achieved if the relative change in the log-likelihood was less than a threshold, which is 10^{-6} by default. This process was repeated 15 times, and the parameter vector with the highest log-likelihood was regarded as the solution. For easier comparison, we also assumed a common covariance matrix among the different Gaussian components. For the algorithm `Basu`, the authors provided their own initialization strategy, which was based on the set of constraints provided. The algorithm was run 15 times, and the solution with the best objective function was picked. The algorithm `Basu` requires a constraint penalty parameter. In our experiment, a wide range of values were tried: 1, 2, 4, 8, 16, 32, 64, 128, 256, 500, 1000, 2000, 4000, 8000, 16000. We only report their results with the best possible penalty values. As a result, the performance of `Basu` reported here might be inflated.

5.6.2.3 Performance criteria

A clustering under constraints algorithm is said to perform well on a data set if the clusters obtained are similar to the ground-truth classes. Consider the k by k "contingency matrix" $\{\tilde{c}_{ij}\}$, where $\tilde{c}_{ij}$ denotes the number of data points that are originally from the i-th class and are assigned to the j-th cluster. If the clusters match the true classes perfectly, there should only be one non-zero entry in each row and each column of the contingency matrix.

Following the common practice in the literature, we summarize the contingency matrix by the F-score and the normalized mutual information (NMI). Consider the "recall matrix" $\{\tilde{r}_{ij}\}$ in which the entries are defined by $\tilde{r}_{ij} = \tilde{c}_{ij} / \sum_{j'} \tilde{c}_{ij'}$. Intuitively, $\tilde{r}_{ij}$ denotes the proportion of the i-th class that is "recalled" by the j-th cluster. The "precision matrix" $\{\tilde{p}_{ij}\}$, on the other hand, is defined by $\tilde{p}_{ij} = \tilde{c}_{ij} / \sum_{i'} \tilde{c}_{i'j}$. It represents how "pure" the j-th cluster is with respect to the i-th class. Entries in the F-score matrix $\{\tilde{f}_{ij}\}$ are simply the harmonic mean of the corresponding entries in the precision and recall matrices, i.e., $\tilde{f}_{ij} = 2\tilde{r}_{ij}\tilde{p}_{ij}/(\tilde{r}_{ij} + \tilde{p}_{ij})$. The F-score of the i-th class, $\tilde{F}_i$, is obtained by assuming that the i-th class matches[12] with the best cluster, i.e., $\tilde{F}_i = \max_j \tilde{f}_{ij}$. The overall F-score is computed as the weighted sum of the individual $\tilde{F}_i$ according to the sizes of the true classes, i.e.,

$$\text{F-score} = \sum_{i=1}^{k} \frac{\sum_{j=1}^{k} \tilde{c}_{ij}}{n} \tilde{F}_i \tag{5.34}$$

Note that the precision of an empty cluster is undefined. This problem can be circumvented if we restrict that empty clusters, if any, should not contribute to the overall F-score.

The computation of normalized mutual information interprets the true class label and the cluster label as two random variables U and V. The contingency table, after dividing by n (the number of objects), forms the joint distribution of U and V. The mutual information (MI) between U and V can be computed based on the joint distribution. Since the range of the mutual information depends on the sizes of the true classes and the sizes of the clusters, we normalize the MI by the average of the entropies of U and V (denoted by $H(U)$ and $H(V)$) so that the resulting value lies between zero

[12]Here, we do not require that one cluster can only match to one class. If this one-to-one correspondence is desired, the Hungarian algorithm should be used to perform the matching instead of the max operation to compute $\tilde{F}_i$.

and one. Formally, we have

$$
\begin{aligned}
H(U) &= -\sum_{i=1}^{k} \frac{\sum_{j=1}^{k} \tilde{c}_{ij}}{n} \log \frac{\sum_{j=1}^{k} \tilde{c}_{ij}}{n} \\
H(V) &= -\sum_{j=1}^{k} \frac{\sum_{i=1}^{k} \tilde{c}_{ij}}{n} \log \frac{\sum_{i=1}^{k} \tilde{c}_{ij}}{n} \\
H(U,V) &= -\sum_{i=1}^{k} \sum_{j=1}^{k} \frac{\tilde{c}_{ij}}{n} \log \frac{\tilde{c}_{ij}}{n} \\
\text{MI} &= H(U) + H(V) - H(U,V) \\
\text{NMI} &= \frac{\text{MI}}{(H(U)+H(V))/2}.
\end{aligned}
\tag{5.35}
$$

For both F-score and NMI, the higher the value, the better the match between the clusters and the true classes. For a perfect match, both NMI and F-score take the value of 1. When the cluster labels are completely independent of the class labels, NMI takes its smallest value of 0. The minimum value of F-score depends on the sizes of the true classes. If all the classes are of equal sizes, the lower bound of F-score is $1/k$. In general, the lower bound of F-score is higher, and it can be more than 0.5 if there is a dominant class.

5.6.2.4 Results

The results of clustering the data sets mentioned in Section 5.6.2.1 when there are no constraints are shown in Table 5.3. In the absence of constraints, both the proposed algorithm and `Shental` effectively find the cluster parameter vector that maximizes the log-likelihood, whereas `Basu` is the same as the k-means algorithm. One may be surprised to discover from Table 5.3 that even though the proposed algorithm and `Shental` optimize the same objective function, their results are different. This is understandable when we notice that the line-search Newton algorithm used by the proposed approach and the EM algorithm used by `Shental` can locate different local optima. It is sometimes argued that maximizing the mixture log-likelihood globally is inappropriate as it can go to infinity when one of the Gaussian components has an almost singular covariance matrix. However, this is not the case here, because the covariance matrices all have small condition numbers as seen in Table 5.3. Therefore, among the two solutions produced by the proposed approach and by `Shental`, we take the one with the larger log-likelihood. In the remaining experiments, the no-constraint solutions found by the proposed algorithm were also used as the initial value for `Shental`. It is because we are interested in locating the best possible local optima for the objective functions.

The results of running our proposed algorithm, `Shental`, and `Basu`, with 1% constraint level, 2% constraint level, 3% constraint level, 5% constraint level, 10% constraint level, and 15% constraint level are shown in Tables 5.4, 5.5, 5.6, 5.7, 5.8, and 5.9, respectively. In these tables, the columns under "Proposed" correspond to the performance of the proposed algorithm. The heading λ denotes the value of the constraint strength as determined by the validation procedure. The heading "`Shental`, default init" corresponds to the performance when the algorithm `Shental` is initialized by its default strategy, whereas "`Shental`, special init" corresponds to the result when `Shental` is initialized by the no-constraint solution found by the proposed approach. The heading "log-lik" shows the log-likelihood of the resulting parameter vector. Among these two solutions of `Shental`, the one with a higher log-likelihood is selected, and its performance is shown under the

		Proposed				Shental				Basu	
	n	F	NMI	log-lik	K	F	NMI	log-lik	K	F	NMI
derm	366	0.817	0.868	-4.608×10^3	9.4×10^0	0.813	0.868	-4.608×10^3	9.5×10^0	0.836	0.880
digits	5620	0.755	0.745	-6.153×10^5	3.9×10^1	0.756	0.745	-6.153×10^5	3.9×10^1	0.825	0.771
ion	351	0.652	0.066	-3.579×10^3	1.5×10^1	0.652	0.066	-3.579×10^3	1.5×10^1	0.718	0.135
mfeat-fou	2000	0.721	0.697	3.062×10^4	8.9×10^0	0.698	0.675	3.059×10^4	8.2×10^0	0.745	0.688
UCI-seg	2310	0.581	0.523	-2.504×10^4	7.7×10^2	0.716	0.690	-2.598×10^4	3.1×10^1	0.704	0.688
wdbc	569	0.685	0.005	-1.044×10^4	8.9×10^1	0.677	0.001	-1.053×10^4	8.8×10^1	0.916	0.583
austra	690	0.523	0.001	-5.506×10^3	6.8×10^2	0.523	0.001	-5.147×10^3	1.9×10^3	0.520	0.000
german	1000	0.649	0.008	-2.492×10^3	3.9×10^3	0.567	0.001	-2.438×10^3	4.0×10^3	0.639	0.015
heart	270	0.590	0.024	-1.674×10^3	1.5×10^2	0.590	0.024	-1.674×10^3	1.5×10^2	0.586	0.022
sat	6435	0.715	0.628	-6.627×10^5	8.4×10^2	0.717	0.632	-6.627×10^5	8.4×10^2	0.716	0.616
vehicle	846	0.510	0.250	-7.012×10^3	3.9×10^2	0.475	0.217	-7.078×10^3	3.9×10^2	0.413	0.111
script	12938	0.630	0.490	-1.610×10^5	1.5×10^1	0.552	0.352	-1.606×10^5	1.3×10^1	0.717	0.553
texture	4000	0.977	0.913	-6.484×10^4	9.5×10^1	0.977	0.913	-6.484×10^4	9.5×10^1	0.956	0.861
ethn	2630	0.652	0.008	1.503×10^5	5.5×10^1	0.652	0.008	1.503×10^5	5.5×10^1	0.759	0.203
Mondrian	10201	0.810	0.801	4.169×10^4	1.6×10^4	0.810	0.801	4.169×10^4	1.6×10^4	0.766	0.793
diff-300	300	0.496	0.058	3.365×10^3	1.1×10^1	0.499	0.131	3.278×10^3	9.3×10^0	0.707	0.454
sim-300	291	0.495	0.029	3.468×10^3	5.8×10^1	0.495	0.029	3.468×10^3	5.8×10^1	0.474	0.033
same-300	297	0.492	0.042	3.225×10^3	6.5×10^0	0.493	0.051	3.307×10^3	7.8×10^0	0.478	0.076

Table 5.3: Performance of different clustering algorithms in the absence of constraints. Both the proposed algorithm and Shental maximize the log-likelihood in this case, though the former uses the line-search Newton whereas the latter uses the EM algorithm. The headings n, F, NMI, log-lik and K denote the number of data points, the F-score, the normalized mutual information, the log-likelihood, and the condition number of the common covariance matrix, respectively.

heading "Shental, combine".

From these tables, we can see that Shental with default initialization often yields a higher performance than Shental with the special initialization. However, the log-likelihood of Shental with default initialization is sometimes smaller. By the principle of maximum likelihood, such a solution, though it has a higher F-score and/or NMI, should not be accepted. This observation has the implication that the good performance of Shental as reported in comparative work such as in [161] might be due to the initialization strategy instead of the model used. The fact that we are more interested in comparing the model used in Shental with that used in the proposed approach, instead of the strategy for initialization, is the reason why we run Shental with the special initialization. We have also tried to do something similar with Basu, but its initialization routine is integrated with the main clustering routine so that it is non-trivial to modify the initialization strategy.

The numbers listed in Tables 5.3 to 5.9 are visualized in Figures 5.7 to 5.13. For each data set, we draw the F-score and the NMI with an increasing number of constraints. The horizontal axis corresponds to different constraint levels in terms of the percentages of the number of data points, whereas the vertical axis corresponds to the F-score or the NMI. The results of the proposed algorithm, Shental, and Basu are shown by the (red) solid lines, (blue) dotted lines, and (black) dashed lines, respectively. For comparison, the (gray) dashdot lines in the figures show the F-score and the NMI due to a classifier trained using the labels of all the objects in the data set under the assumption that the class conditional densities are Gaussian with a common covariance matrix. The data sets are grouped according to the performance of the proposed algorithms. The proposed algorithm outperformed both Shental and Basu for the data sets shown in Figures 5.7 to 5.9. The performance of the proposed algorithm is comparable to its competitors for the data sets shown in Figures 5.10 to 5.12. For the data sets shown in Figures 5.13, the proposed algorithm is slightly inferior to one of its competitors. We shall examine the performance on individual data sets later.

Perhaps the first observation from these figures is that the performance is not monotonic, i.e., the F-score and the NMI can actually decrease when there are additional constraints. This is counter-intuitive, because one expects improved results when more information (in the form of constraints) is fed as the input to the algorithms. Note that this lack of monotonicity is observed for all the three algorithms. There are three reasons for this. First, the additional constraints can be based on data points that are erroneously labeled (errors in the ground truth), or they are "outlier" in the sense that they would be mis-classified by most reasonable supervised classifiers trained with all the labels known. The additional constraints in this case serve as "mis-information", and it can hurt the performance of the clustering under constraints algorithms. This effect is more severe for the proposed approach when there are only a small number of constraints, because the influence of each of the constraints may be magnified by a large value of λ. The second reason is that an algorithm may locate a poor local optima. In general, the larger the number of constraints, the greater the number of local optima in the energy landscape. So, the proposed algorithm as well as Shentaland Basu is more likely to get trapped in poor local optima. This trend is the most obvious for Basu, as the performance at 10% and 15% constraint levels dropped for more than half of the data sets. This is not surprising, because the iterative conditional mode used by Basu is greedy and it is likely to get trapped in local optima. The third reason is specific to the proposed approach. It is due to the random nature of the partitioning of the constraints into training set and validation set. If we have an unfavorable split, the value of λ found by minimizing the number of violations on the set of validation constraints can be suboptimal. In fact, we observe that whenever there is a significant drop in the F-score and NMI, there often exists a better value of λ than the one found by the validation procedure.

Performance on Individual Data Sets The result on the ethn data set can be seen in Figures 5.7(a) and 5.7(b). The performance of the proposed algorithm improves with additional constraints, and it outperforms Shental and Basu at all constraint levels. A similar phenomenon occurs for the Mondrian data set (Figures 5.7(c) and 5.7(d)) and the ion data set (Figures 5.7(e) and 5.7(f)). For Mondrian, note that 1% constraint level is already sufficient to bias the cluster parameter to match the result using the ground-truth labels. Additional constraints only help marginally. The performance of the proposed algorithm for the script data set (Figures 5.8(a) and 5.8(b)) is better than Shental and Basu for all constraint levels except 1%, where the proposed algorithm is inferior to the result of Basu. However, given how much better the k-means algorithm is when compared with the EM algorithm in the absence of constraints, it is fair to say that the proposed algorithm is doing a decent job. For the data set derm, the clustering solution without any constraints is pretty good: that solution, in fact, satisfies all the constraints when the constraint levels are 1% and 2%. Therefore, it is natural that the performance does not improve with the provision of the constraints. However, when the constraint level is higher than 2%, the proposed algorithm again outperforms Shental and Basu (Figures 5.8(c) and 5.8(d)). The performance of the proposed algorithm on the vehicle data set is superior to Shental and Basu for all constraint levels except 5%, where the performance of Shental is slightly superior. For the data set wdbc, the performance of the proposed algorithm (Figures 5.9(a) and 5.9(b)) is better than Shental at all constraint levels except 5%. The proposed algorithm outperforms Basu when the constraint level is higher than 1%.

The F-score of the proposed algorithm on the UCI-seg data (Figures 5.10(a)) is superior to Shental at three constraint levels and is superior to Basu at all but 1% constraint level. On the other hand, if NMI is used (Figure 5.10(b)), the proposed algorithm does not do as well as the others. For the heart data set, the proposed algorithm is superior to Shental at all constraint levels, but it is superior to Basu at only 3% constraint level (Figures 5.10(c) and 5.10(d)). Note that the performance of Basu might be inflated because we only report its best results among all possible values of constraint penalty in this algorithm. We can regard the performance of the proposed algorithm on the austra data set (Figures 5.10(e) and 5.10(f)) as a tie with Shental and Basu, because the proposed algorithm outperforms Shental and Basuat three out of six possible constraint levels. For the german data set, the proposed algorithm performs the best in terms of NMI (Figure 5.11(b)), though the performances of all three algorithms are not that good. Apparently, this is a difficult data set. The performance of the proposed algorithm is less impressive when F-score is used, however (Figure 5.11(a)). The proposed algorithm is superior to Shental in performance for the sim-300 data set (Figures 5.11(c) and 5.11(d)). While the proposed algorithm has a tie in performance when compared with Basu based on the F-score, Basu outperforms the proposed algorithm on this data set when NMI is used. The result of the diff-300 data set (Figures 5.11(e) and 5.11(f)) is somewhat similar: the proposed algorithm outperforms Shental at all constraint levels, but it is inferior to Basu. Given the fact that the k-means algorithm is much better than EM in the absence of constraints for this data set, the proposed algorithm is not as bad as it first seems. For the sat data set (Figures 5.12(a) and 5.12(b)), the proposed algorithm outperforms Shental and Basu significantly in terms of F-score when the constraint levels are 10% and 15%. The improvement in NMI is less significant, though the proposed method is still the best at three constraint levels. The result of the digits data set (Figures 5.12(c) and 5.12(d)) is similar: the proposed method is superior to its competitors at three and four constraint levels if F-score and NMI are used as the evaluation criteria, respectively.

It is difficult to draw any conclusion on the performance of the three algorithms on the mfeat-fou data set (Figures 5.13(a) and 5.13(b)). The performances of all three algorithms go up and down with an increasing number of constraints. Apparently this data set is fairly noisy, and clustering with constraints is not appropriate for this data set. For the data set `same-300`, the proposed algorithm does not perform well: it has a tie with `Shental`, but it is inferior to `Basu` at all constraint levels, as seen in Figures 5.13(c) and 5.13(d). The performance of the proposed algorithm is better than `Shental` only at the 15% constraint level for the data set `texture` (Figures 5.13(e) and 5.13(f)). The proposed algorithm is superior to `Basu` for this data set, though this is probably due to the better performance of the EM algorithm in the absence of constraints. Note that this data set is a relatively easy data set for model-based clustering: both k-means and EM have a F-score higher than 0.95 when no constraints are used.

5.6.3 Experiments on Feature Extraction

We have also tested the idea of learning the low-dimensional subspace and the clusters simultaneously in the presence of constraints. Our first experiment in this regard is based on the data set shown in Figure 5.3. The two features were standardized to variance one before applying the algorithm described in Section 5.5 with the two must-link constraints. Based on the result shown in Figure 5.14(a), we can see that a good projection direction was found by the proposed algorithm. The projected data follow the Gaussian distribution well, as evident from Figure 5.14(b).

Our second experiment is about the combination of feature extraction and the kernel trick to detect clusters with general shapes. The two-ring data set (Figure 5.15(a)) considered in [158], which used a hidden Markov random field approach for clustering with constraints in kernel k-means, was used. As in [158], we applied the RBF kernel to transform this data set of 200 points nonlinearly. The kernel width was set to 0.2, which was the 20-percentile of all the pairwise distances. Unlike [158], we applied kernel PCA to this data set and extracted 20 features. The algorithm described in Section 5.5 was used to learn a good projection of these 20 features into a 2D space while clustering the data into two groups simultaneously in the presence of 60 randomly generated constraints. The result shown in Figure 5.15(b) indicates that the algorithm successfully found a 2D subspace such that the two clusters were Gaussian-like, and all the constraints were satisfied. When we plot the cluster labels of the original two-ring data set, we can see that the desired clusters (the "inner" and the "outer" rings) were recovered perfectly (Figure 5.15(c)). Note that the algorithm described in [158] required at least 450 constraints to identify the two clusters perfectly, whereas we have only used 60 constraints. For comparison, the spectral clustering algorithm in [194] was applied to this data set using the same kernel matrix as the similarity. The two desired clusters could not be recovered (Figure 5.15(d)). In fact, the two desired clusters were never recovered even when we tried other values of kernel widths.

5.7 Discussion

5.7.1 Time Complexity

The computation of the objective function and its gradient requires the calculation of $\tilde{r}_{ij}$, s_{ij}, w_{ij}, and the weighted sum of different sufficient statistics with r_{ij} and w_{ij} as weights. When compared with the EM algorithm for standard model-based clustering, the extra computation by the proposed algorithm is due to s_{ij}, w_{ij}, and the accumulation of the corresponding sufficient statistics. These take $O(kd(m^+ + m^- + n^*))$ time, where k, d, m^+, m^-, n^* denote the number of clusters, the dimension of the feature vector, the number of must-link constraints, the number of must-not-link constraints, and the number of data points involved in any constraint, respectively. This is smaller than the $O(kdn)$ time required for one iteration of the EM algorithm, with n indicating the total

			Proposed			`Shental`, default init.			`Shental`, special init.			`Shental`, combined		`Basu`	
	n	m	F	NMI	λ	F	NMI	log-lik	F	NMI	log-lik	F	NMI	F	NMI
`derm`	366	4	0.817	0.868	0	0.813	0.868	-4.607×10^3	0.817	0.868	-4.607×10^3	0.817	0.868	0.838	0.880
`digits`	5620	56	0.790	0.743	1.0×10^3	0.781	0.779	-6.157×10^5	0.848	0.818	-6.164×10^5	0.781	0.779	0.814	0.757
`ion`	351	4	0.812	0.285	3.2×10^2	0.651	0.046	-3.579×10^3	0.651	0.046	-3.579×10^3	0.651	0.046	0.720	0.138
`mfeat-fou`	2000	20	0.678	0.664	3.2×10^2	0.761	0.701	3.055×10^4	0.676	0.661	3.052×10^4	0.761	0.701	0.749	0.694
`UCI-seg`	2310	23	0.714	0.672	1.0×10^2	0.707	0.662	-2.643×10^4	0.712	0.685	-2.552×10^4	0.712	0.685	0.711	0.704
`wdbc`	569	6	0.677	0.175	1.0×10^2	0.782	0.354	-1.058×10^4	0.672	0.141	-1.056×10^4	0.672	0.141	0.907	0.551
`austra`	690	7	0.546	0.000	3.2×10^4	0.854	0.421	-6.217×10^3	0.601	0.025	-6.086×10^3	0.601	0.025	0.537	0.003
`german`	1000	10	0.649	0.008	0	0.567	0.001	-4.033×10^3	0.648	0.008	-4.232×10^3	0.567	0.001	0.653	0.000
`heart`	270	3	0.762	0.205	3.2×10^2	0.590	0.024	-1.674×10^3	0.590	0.024	-1.674×10^3	0.590	0.024	0.830	0.339
`sat`	6435	64	0.711	0.605	3.2×10^2	0.716	0.632	-6.628×10^5	0.716	0.632	-6.628×10^5	0.716	0.632	0.714	0.614
`vehicle`	846	8	0.506	0.218	1.0×10^3	0.484	0.223	-7.674×10^3	0.487	0.185	-7.099×10^3	0.487	0.185	0.420	0.115
`script`	12938	129	0.618	0.503	1.0×10^3	0.670	0.518	-1.617×10^5	0.503	0.321	-1.616×10^5	0.503	0.321	0.659	0.535
`texture`	4000	40	0.977	0.913	0	0.977	0.915	-6.483×10^4	0.977	0.913	-6.483×10^4	0.977	0.913	0.959	0.866
`ethn`	2630	26	0.831	0.435	1.0×10^3	0.651	0.008	1.502×10^5	0.651	0.008	1.502×10^5	0.651	0.008	0.571	0.000
`Mondrian`	10201	102	0.966	0.897	3.2×10^2	0.809	0.798	4.146×10^4	0.799	0.793	4.131×10^4	0.809	0.798	0.766	0.787
`diff-300`	300	3	0.566	0.295	3.2×10^2	0.494	0.144	3.244×10^3	0.489	0.040	3.253×10^3	0.489	0.040	0.731	0.483
`sim-300`	291	3	0.562	0.109	3.2×10^2	0.496	0.042	3.301×10^3	0.493	0.023	3.462×10^3	0.493	0.023	0.507	0.108
`same-300`	297	3	0.580	0.164	3.2×10^2	0.585	0.179	3.135×10^3	0.489	0.052	3.151×10^3	0.489	0.052	0.594	0.183

Table 5.4: Performance of clustering under constraints algorithms when the constraint level is 1%. The headings n, m, F, NMI, λ, and log-lik denote the number of data points, the number of constraints, the F-score, the normalized mutual information, the optimal λ for the proposed algorithm found by the validation procedure, and the log-likelihood, respectively.

			Proposed			`Shental`, default init.			`Shental`, special init.			`Shental`, combined		`Basu`	
	n	m	F	NMI	λ	F	NMI	log-lik	F	NMI	log-lik	F	NMI	F	NMI
`derm`	366	7	0.817	0.868	0	0.817	0.868	-4.605×10^3	0.817	0.868	-4.605×10^3	0.817	0.868	0.838	0.880
`digits`	5620	112	0.747	0.710	1.0×10^6	0.756	0.745	-6.153×10^5	0.747	0.721	-6.155×10^5	0.756	0.745	0.815	0.760
`ion`	351	7	0.755	0.175	3.2×10^2	0.651	0.046	-3.579×10^3	0.651	0.046	-3.579×10^3	0.651	0.046	0.721	0.140
`mfeat-fou`	2000	40	0.737	0.678	3.2×10^2	0.701	0.676	3.059×10^4	0.756	0.705	3.059×10^4	0.701	0.676	0.752	0.700
`UCI-seg`	2310	46	0.721	0.709	1.0×10^4	0.702	0.667	-2.645×10^4	0.711	0.683	-2.577×10^4	0.711	0.683	0.623	0.577
`wdbc`	569	11	0.913	0.636	3.2×10^2	0.671	0.151	-1.056×10^4	0.681	0.002	-1.058×10^4	0.671	0.151	0.909	0.556
`austra`	690	14	0.616	0.058	3.2×10^2	0.523	0.001	-6.113×10^3	0.601	0.025	-6.285×10^3	0.523	0.001	0.586	0.021
`german`	1000	20	0.631	0.011	1.0×10^4	0.651	0.001	-4.395×10^3	0.648	0.008	-4.368×10^3	0.648	0.008	0.579	0.005
`heart`	270	5	0.762	0.205	1.0×10^2	0.594	0.026	-1.772×10^3	0.594	0.026	-1.772×10^3	0.594	0.026	0.830	0.339
`sat`	6435	129	0.719	0.637	3.2×10^1	0.717	0.634	-6.627×10^5	0.715	0.630	-6.627×10^5	0.715	0.630	0.713	0.613
`vehicle`	846	17	0.737	0.562	1.0×10^3	0.485	0.183	-7.137×10^3	0.517	0.261	-7.079×10^3	0.517	0.261	0.418	0.114
`script`	12938	259	0.735	0.576	1.0×10^3	0.671	0.518	-1.617×10^5	0.670	0.518	-1.617×10^5	0.670	0.518	0.654	0.536
`texture`	4000	80	0.976	0.913	3.2×10^1	0.978	0.918	-6.482×10^4	0.978	0.918	-6.482×10^4	0.978	0.918	0.958	0.865
`ethn`	2630	53	0.924	0.613	3.2×10^2	0.645	0.000	1.498×10^5	0.648	0.006	1.500×10^5	0.648	0.006	0.572	0.001
`Mondrian`	10201	204	0.966	0.898	3.2×10^2	0.809	0.797	4.135×10^4	0.809	0.797	4.135×10^4	0.809	0.797	0.766	0.785
`diff-300`	300	6	0.616	0.199	3.2×10^2	0.480	0.091	3.157×10^3	0.492	0.070	3.177×10^3	0.492	0.070	0.741	0.494
`sim-300`	291	6	0.532	0.071	3.2×10^2	0.495	0.029	3.468×10^3	0.493	0.017	3.137×10^3	0.495	0.029	0.515	0.137
`same-300`	297	6	0.446	0.025	1.0×10^2	0.586	0.182	3.091×10^3	0.493	0.026	3.109×10^3	0.493	0.026	0.598	0.188

Table 5.5: Performance of clustering under constraints algorithms when the constraint level is 2%. The headings n, m, F, NMI, λ, and log-lik denote the number of data points, the number of constraints, the F-score, the normalized mutual information, the optimal λ for the proposed algorithm found by the validation procedure, and the log-likelihood, respectively.

			Proposed			Shental, default init.			Shental, special init.			Shental, combined		Basu	
	n	m	F	NMI	λ	F	NMI	log-lik	F	NMI	log-lik	F	NMI	F	NMI
derm	366	11	0.951	0.914	3.2×10^2	0.813	0.868	-4.605×10^3	0.817	0.868	-4.605×10^3	0.817	0.868	0.837	0.874
digits	5620	169	0.827	0.790	1.0×10^3	0.758	0.746	-6.153×10^5	0.754	0.744	-6.153×10^5	0.754	0.744	0.815	0.755
ion	351	11	0.724	0.130	3.2×10^3	0.651	0.032	-3.578×10^3	0.651	0.068	-3.608×10^3	0.651	0.032	0.715	0.132
mfeat-fou	2000	60	0.721	0.697	0	0.717	0.677	3.052×10^4	0.720	0.697	3.061×10^4	0.720	0.697	0.684	0.657
UCI-seg	2310	69	0.695	0.621	3.2×10^5	0.688	0.644	-2.668×10^4	0.715	0.689	-2.617×10^4	0.715	0.689	0.648	0.638
wdbc	569	17	0.924	0.668	3.2×10^1	0.907	0.609	-1.059×10^4	0.678	0.002	-1.058×10^4	0.678	0.002	0.909	0.556
austra	690	21	0.640	0.069	3.2×10^2	0.855	0.428	-6.306×10^3	0.523	0.001	-6.262×10^3	0.523	0.001	0.571	0.013
german	1000	30	0.591	0.012	3.2×10^3	0.566	0.001	-4.620×10^3	0.602	0.001	-4.521×10^3	0.602	0.001	0.578	0.005
heart	270	8	0.762	0.205	1.0×10^2	0.594	0.027	-1.827×10^3	0.592	0.067	-1.838×10^3	0.594	0.027	0.549	0.012
sat	6435	193	0.712	0.585	1.0×10^3	0.715	0.630	-6.627×10^5	0.715	0.630	-6.627×10^5	0.715	0.630	0.716	0.614
vehicle	846	25	0.523	0.257	3.2×10^3	0.448	0.200	-7.835×10^3	0.383	0.070	-7.156×10^3	0.383	0.070	0.419	0.113
script	12938	388	0.746	0.563	3.2×10^3	0.671	0.517	-1.617×10^5	0.670	0.517	-1.617×10^5	0.670	0.517	0.654	0.535
texture	4000	120	0.978	0.918	1.0×10^2	0.978	0.918	-6.481×10^4	0.978	0.917	-6.481×10^4	0.978	0.918	0.960	0.867
ethn	2630	79	0.956	0.739	1.0×10^2	0.645	0.000	1.498×10^5	0.650	0.008	1.499×10^5	0.650	0.008	0.646	0.068
Mondrian	10201	306	0.967	0.900	3.2×10^2	0.809	0.796	4.122×10^4	0.809	0.796	4.122×10^4	0.809	0.796	0.774	0.753
diff-300	300	9	0.643	0.254	3.2×10^2	0.481	0.074	3.174×10^3	0.512	0.175	3.268×10^3	0.512	0.175	0.629	0.382
sim-300	291	9	0.473	0.039	3.2×10^2	0.488	0.112	3.250×10^3	0.485	0.046	3.035×10^3	0.488	0.112	0.574	0.180
same-300	297	9	0.527	0.080	1.0×10^2	0.477	0.027	3.097×10^3	0.489	0.039	3.129×10^3	0.489	0.039	0.618	0.222

Table 5.6: Performance of clustering under constraints algorithms when the constraint level is 3%. The headings n, m, F, NMI, λ, and log-lik denote the number of data points, the number of constraints, the F-score, the normalized mutual information, the optimal λ for the proposed algorithm found by the validation procedure, and the log-likelihood, respectively.

			Proposed			`Shental`, default init.			`Shental`, special init.			`Shental`, combined		`Basu`	
	n	m	F	NMI	λ	F	NMI	log-lik	F	NMI	log-lik	F	NMI	F	NMI
`derm`	366	18	0.954	0.918	1.0×10^2	0.946	0.912	-4.669×10^3	0.817	0.868	-4.604×10^3	0.817	0.868	0.838	0.874
`digits`	5620	281	0.731	0.707	3.2×10^1	0.779	0.765	-6.154×10^5	0.712	0.717	-6.162×10^5	0.779	0.765	0.725	0.711
`ion`	351	18	0.823	0.367	3.2×10^1	0.650	0.034	-3.594×10^3	0.650	0.034	-3.594×10^3	0.650	0.034	0.710	0.126
`mfeat-fou`	2000	100	0.729	0.681	1.0×10^3	0.757	0.706	3.059×10^4	0.768	0.722	3.059×10^4	0.757	0.706	0.746	0.671
`UCI-seg`	2310	116	0.772	0.721	1.0×10^3	0.714	0.688	-2.617×10^4	0.713	0.689	-2.553×10^4	0.713	0.689	0.645	0.628
`wdbc`	569	28	0.879	0.509	3.2×10^1	0.894	0.574	-1.059×10^4	0.892	0.570	-1.059×10^4	0.894	0.574	0.875	0.444
`austra`	690	35	0.796	0.269	1.0×10^3	0.613	0.000	-6.226×10^3	0.855	0.428	-6.303×10^3	0.613	0.000	0.848	0.409
`german`	1000	50	0.573	0.000	3.2×10^3	0.566	0.001	-4.788×10^3	0.665	0.000	-4.963×10^3	0.566	0.001	0.586	0.000
`heart`	270	14	0.838	0.366	3.2×10^2	0.594	0.027	-1.894×10^3	0.594	0.027	-1.894×10^3	0.594	0.027	0.838	0.364
`sat`	6435	322	0.733	0.624	3.2×10^2	0.719	0.638	-6.627×10^5	0.717	0.631	-6.627×10^5	0.719	0.638	0.715	0.611
`vehicle`	846	42	0.443	0.132	3.2×10^1	0.490	0.251	-7.794×10^3	0.475	0.214	-7.188×10^3	0.475	0.214	0.420	0.113
`script`	12938	647	0.733	0.552	1.0×10^4	0.672	0.518	-1.618×10^5	0.629	0.485	-1.612×10^5	0.629	0.485	0.720	0.551
`texture`	4000	200	0.977	0.914	3.2×10^2	0.978	0.918	-6.480×10^4	0.978	0.918	-6.480×10^4	0.978	0.918	0.959	0.866
`ethn`	2630	132	0.958	0.749	3.2×10^2	0.851	0.408	1.496×10^5	0.646	0.007	1.497×10^5	0.646	0.007	0.738	0.234
`Mondrian`	10201	510	0.969	0.904	3.2×10^3	0.808	0.794	4.101×10^4	0.808	0.794	4.101×10^4	0.808	0.794	0.782	0.751
`diff-300`	300	15	0.664	0.381	1.0×10^4	0.476	0.112	3.119×10^3	0.493	0.048	3.179×10^3	0.493	0.048	0.850	0.623
`sim-300`	291	15	0.526	0.086	3.2×10^4	0.482	0.015	3.246×10^3	0.494	0.034	3.280×10^3	0.494	0.034	0.606	0.253
`same-300`	297	15	0.413	0.033	1.0×10^2	0.477	0.079	3.129×10^3	0.486	0.047	3.103×10^3	0.477	0.079	0.573	0.166

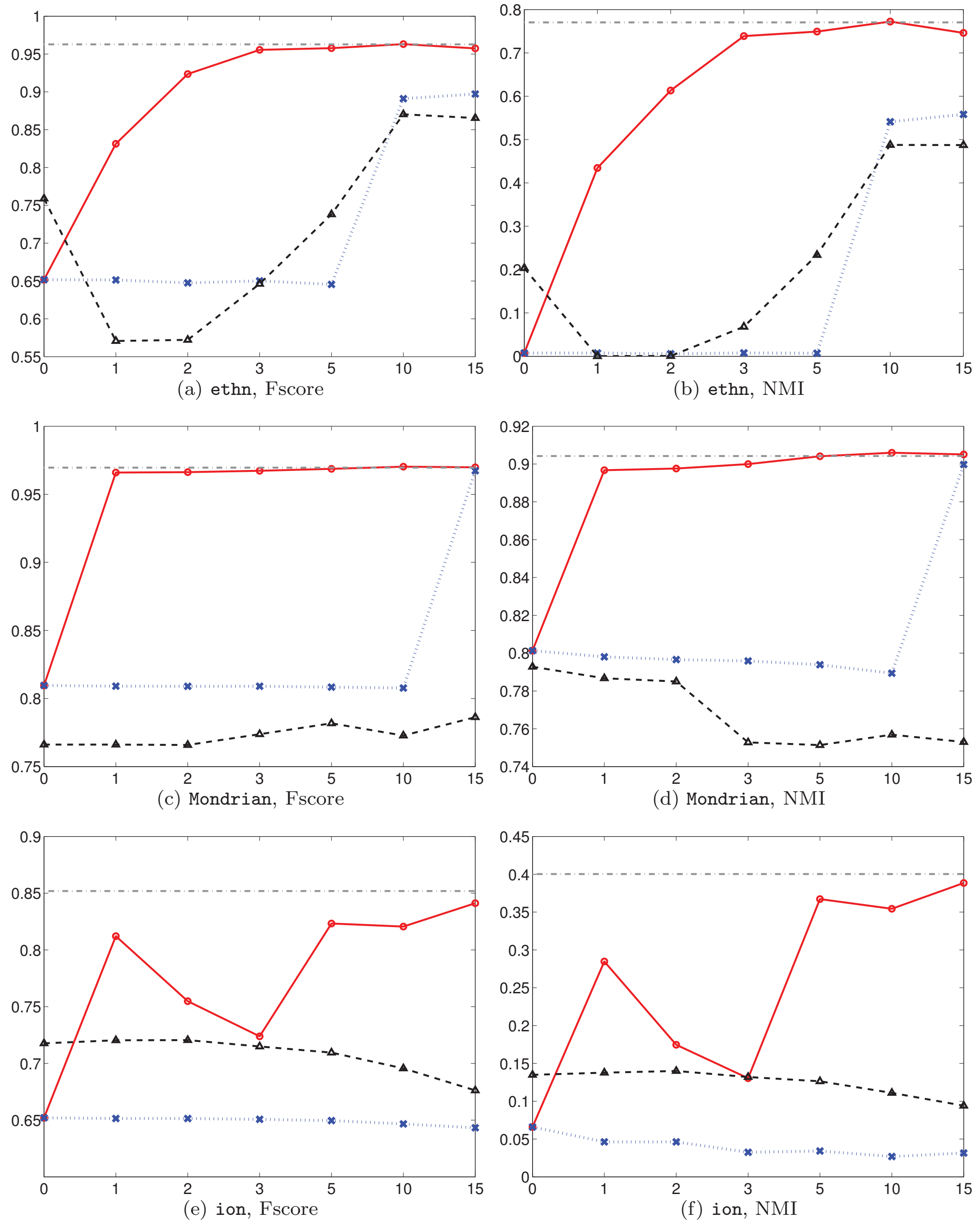
Table 5.7: Performance of clustering under constraints algorithms when the constraint level is 5%. The headings n, m, F, NMI, λ, and log-lik denote the number of data points, the number of constraints, the F-score, the normalized mutual information, the optimal λ for the proposed algorithm found by the validation procedure, and the log-likelihood, respectively.

			Proposed			`Shental`, default init.			`Shental`, special init.			`Shental`, combined		`Basu`	
	n	m	F	NMI	λ	F	NMI	log-lik	F	NMI	log-lik	F	NMI	F	NMI
`derm`	366	37	0.951	0.914	3.2×10^2	0.821	0.869	-4.611×10^3	0.819	0.869	-4.612×10^3	0.821	0.869	0.843	0.890
`digits`	5620	562	0.875	0.813	1.0×10^3	0.780	0.765	-6.155×10^5	0.794	0.751	-6.164×10^5	0.780	0.765	0.737	0.715
`ion`	351	35	0.821	0.354	3.2×10^1	0.647	0.027	-3.637×10^3	0.647	0.027	-3.637×10^3	0.647	0.027	0.696	0.111
`mfeat-fou`	2000	200	0.685	0.658	1.0×10^3	0.761	0.686	3.043×10^4	0.754	0.705	3.058×10^4	0.754	0.705	0.687	0.662
`UCI-seg`	2310	231	0.667	0.571	1.0×10^5	0.692	0.640	-2.661×10^4	0.707	0.683	-2.669×10^4	0.692	0.640	0.638	0.613
`wdbc`	569	57	0.939	0.688	1.0×10^2	0.795	0.376	-1.058×10^4	0.913	0.624	-1.060×10^4	0.795	0.376	0.851	0.383
`austra`	690	69	0.840	0.363	1.0×10^2	0.855	0.426	-6.397×10^3	0.857	0.431	-6.397×10^3	0.855	0.426	0.855	0.430
`german`	1000	100	0.656	0.026	1.0×10^3	0.646	0.008	-5.124×10^3	0.646	0.008	-5.124×10^3	0.646	0.008	0.582	0.006
`heart`	270	27	0.713	0.132	1.0×10^2	0.612	0.037	-1.969×10^3	0.612	0.037	-1.969×10^3	0.612	0.037	0.834	0.359
`sat`	6435	644	0.795	0.658	3.2×10^2	0.718	0.637	-6.627×10^5	0.717	0.631	-6.627×10^5	0.718	0.637	0.719	0.612
`vehicle`	846	85	0.473	0.164	1.0×10^5	0.453	0.212	-7.924×10^3	0.378	0.069	-7.258×10^3	0.378	0.069	0.433	0.117
`script`	12938	1294	0.696	0.518	1.0×10^5	0.673	0.516	-1.620×10^5	0.672	0.515	-1.620×10^5	0.672	0.515	0.687	0.506
`texture`	4000	400	0.973	0.903	3.2×10^{-1}	0.978	0.919	-6.475×10^4	0.976	0.909	-6.475×10^4	0.978	0.919	0.952	0.850
`ethn`	2630	263	0.963	0.772	3.2×10^2	0.891	0.541	1.497×10^5	0.933	0.653	1.495×10^5	0.891	0.541	0.870	0.488
`Mondrian`	10201	1020	0.970	0.906	3.2×10^2	0.808	0.789	4.070×10^4	0.808	0.789	4.070×10^4	0.808	0.789	0.773	0.757
`diff-300`	300	30	0.740	0.409	3.2×10^2	0.607	0.276	3.089×10^3	0.508	0.157	3.065×10^3	0.607	0.276	0.898	0.684
`sim-300`	291	29	0.615	0.201	1.0×10^4	0.569	0.164	3.133×10^3	0.493	0.012	3.145×10^3	0.493	0.012	0.594	0.265
`same-300`	297	30	0.607	0.229	1.0×10^4	0.519	0.112	3.106×10^3	0.545	0.164	3.068×10^3	0.519	0.112	0.641	0.275

Table 5.8: Performance of clustering under constraints algorithms when the constraint level is 10%. The headings n, m, F, NMI, λ, and log-lik denote the number of data points, the number of constraints, the F-score, the normalized mutual information, the optimal λ for the proposed algorithm found by the validation procedure, and the log-likelihood, respectively.

			Proposed			`Shental`, default init.			`Shental`, special init.			`Shental`, combined		`Basu`	
	n	m	F	NMI	λ	F	NMI	log-lik	F	NMI	log-lik	F	NMI	F	NMI
`derm`	366	55	0.954	0.918	1.0×10^2	0.806	0.861	-4.643×10^3	0.819	0.862	-4.634×10^3	0.819	0.862	0.682	0.718
`digits`	5620	843	0.874	0.790	1.0×10^3	0.760	0.746	-6.156×10^5	0.894	0.824	-6.164×10^5	0.760	0.746	0.803	0.749
`ion`	351	53	0.841	0.388	3.2×10^1	0.643	0.032	-3.672×10^3	0.643	0.032	-3.672×10^3	0.643	0.032	0.676	0.094
`mfeat-fou`	2000	300	0.708	0.679	3.2×10^1	0.763	0.705	3.056×10^4	0.709	0.679	3.058×10^4	0.709	0.679	0.673	0.653
`UCI-seg`	2310	347	0.739	0.692	3.2×10^4	0.706	0.661	-2.646×10^4	0.717	0.689	-2.554×10^4	0.717	0.689	0.467	0.395
`wdbc`	569	85	0.972	0.825	3.2×10^1	0.930	0.685	-1.061×10^4	0.930	0.685	-1.061×10^4	0.930	0.685	0.913	0.560
`austra`	690	104	0.851	0.395	3.2×10^3	0.861	0.440	-6.506×10^3	0.861	0.440	-6.506×10^3	0.861	0.440	0.848	0.407
`german`	1000	150	0.649	0.008	0	0.646	0.001	-5.372×10^3	0.648	0.010	-5.283×10^3	0.648	0.010	0.653	0.010
`heart`	270	41	0.819	0.327	1.0×10^3	0.760	0.206	-1.984×10^3	0.609	0.034	-1.977×10^3	0.609	0.034	0.830	0.342
`sat`	6435	965	0.796	0.658	1.0×10^4	0.719	0.638	-6.628×10^5	0.717	0.634	-6.628×10^5	0.719	0.638	0.703	0.610
`vehicle`	846	127	0.562	0.237	3.2×10^3	0.457	0.217	-7.996×10^3	0.392	0.064	-7.399×10^3	0.392	0.064	0.425	0.114
`script`	12938	1941	0.809	0.620	1.0×10^3	0.679	0.517	-1.623×10^5	0.678	0.517	-1.623×10^5	0.679	0.517	0.686	0.501
`texture`	4000	600	0.982	0.927	3.2×10^2	0.979	0.920	-6.470×10^4	0.979	0.920	-6.470×10^4	0.979	0.920	0.957	0.860
`ethn`	2630	395	0.957	0.746	1.0×10^2	0.897	0.558	1.497×10^5	0.957	0.743	1.495×10^5	0.897	0.558	0.865	0.487
`Mondrian`	10201	1530	0.970	0.905	3.2×10^2	0.967	0.900	4.033×10^4	0.967	0.900	4.033×10^4	0.967	0.900	0.786	0.753
`diff-300`	300	45	0.807	0.518	1.0×10^5	0.477	0.065	3.082×10^3	0.480	0.081	3.164×10^3	0.480	0.081	0.937	0.770
`sim-300`	291	44	0.479	0.090	1.0×10^4	0.484	0.075	3.142×10^3	0.490	0.015	3.180×10^3	0.490	0.015	0.602	0.271
`same-300`	297	45	0.436	0.046	1.0×10^4	0.455	0.041	3.078×10^3	0.540	0.113	3.108×10^3	0.540	0.113	0.663	0.340

Table 5.9: Performance of clustering under constraints algorithms when the constraint level is 15%. The headings n, m, F, NMI, λ, and log-lik denote the number of data points, the number of constraints, the F-score, the normalized mutual information, the optimal λ for the proposed algorithm found by the validation procedure, and the log-likelihood, respectively.

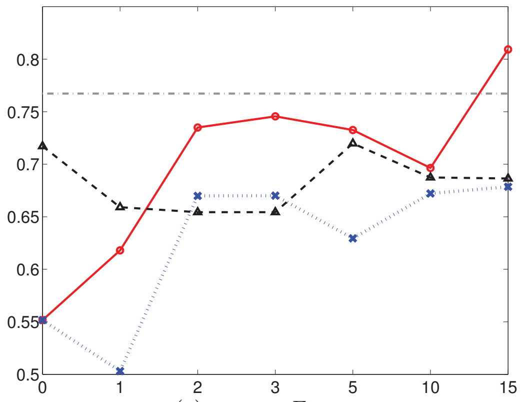


(a) `ethn`, Fscore (b) `ethn`, NMI

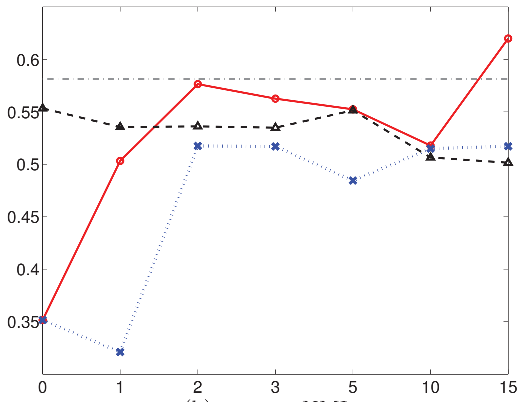
(c) `Mondrian`, Fscore (d) `Mondrian`, NMI

(e) `ion`, Fscore (f) `ion`, NMI

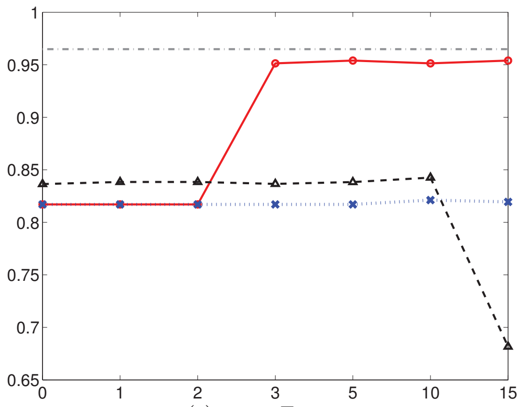
Figure 5.7: F-score and NMI for different algorithms for clustering under constraints for the data sets `ethn`, `Mondrian`, and `ion`. The results of the proposed algorithm, `Shental`, and `Basu` are represented by the red solid line, blue dotted lines and the black dashed line, respectively. The performance of a classifier trained using all the labels is shown by the gray dashdot line. The horizontal axis shows the number of constraints as the percentage of the number of data points.



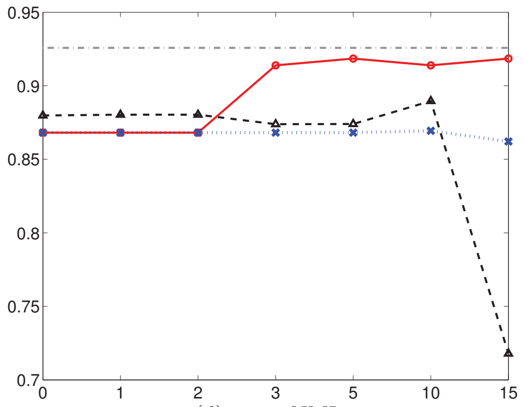
(a) `script`, Fscore



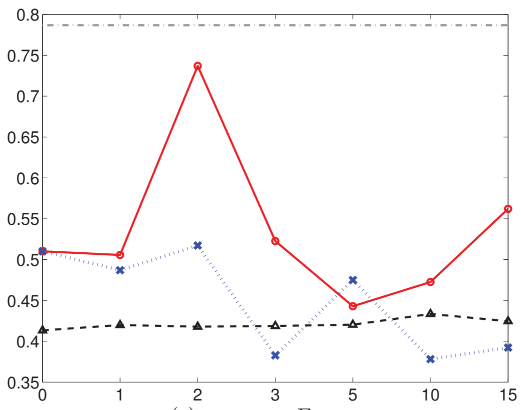
(b) `script`, NMI



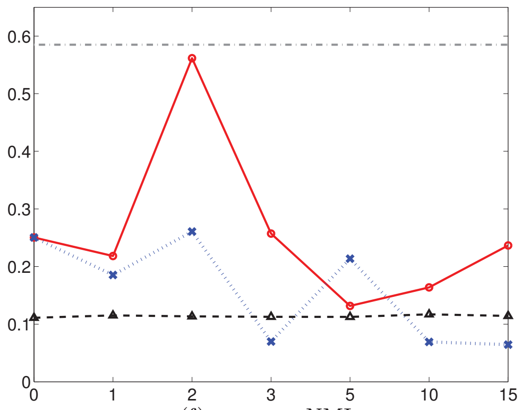
(c) `derm`, Fscore



(d) `derm`, NMI

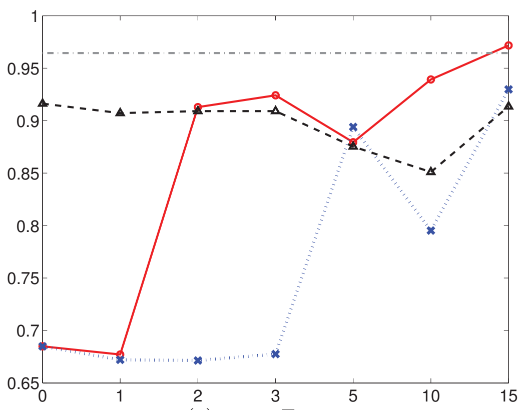


(e) `vehicle`, Fscore

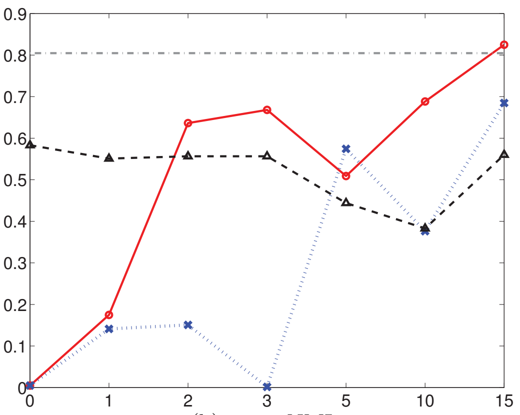


(f) `vehicle`, NMI

Figure 5.8: F-score and NMI for different algorithms for clustering under constraints for the data sets `script`, `derm`, and `vehicle`. The results of the proposed algorithm, `Shental`, and `Basu` are represented by the red solid line, blue dotted lines and the black dashed line, respectively. The performance of a classifier trained using all the labels is shown by the gray dashdot line. The horizontal axis shows the number of constraints as the percentage of the number of data points.

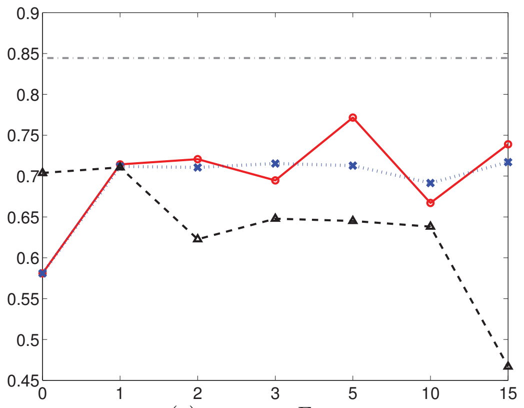


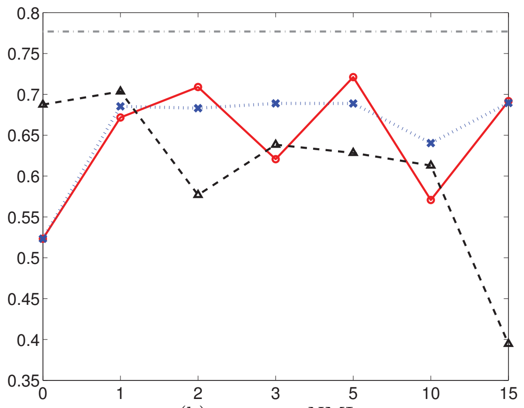
(a) wdbc, Fscore

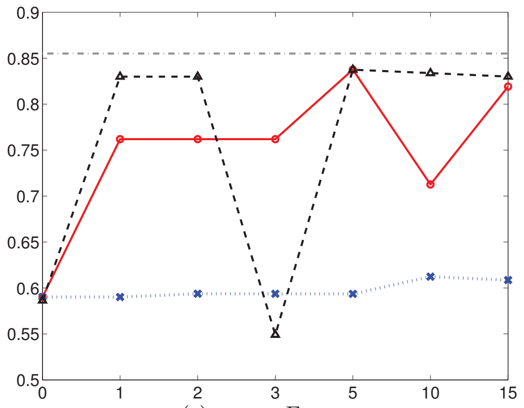


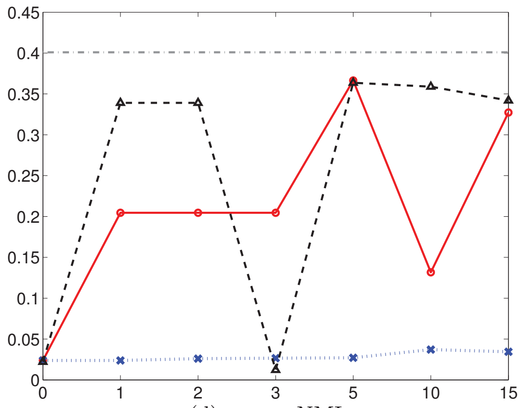
(b) wdbc, NMI

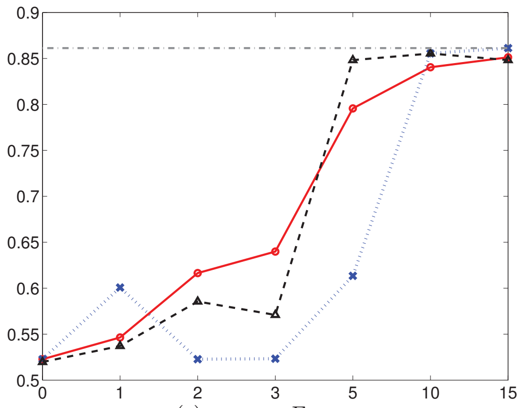
Figure 5.9: F-score and NMI for different algorithms for clustering under constraints for the data sets `wdbc`. The results of the proposed algorithm, `Shental`, and `Basu` are represented by the red solid line, blue dotted lines and the black dashed line, respectively. The performance of a classifier trained using all the labels is shown by the gray dashdot line. The horizontal axis shows the number of constraints as the percentage of the number of data points.


(a) `UCI-seg`, Fscore


(b) `UCI-seg`, NMI


(c) `heart`, Fscore


(d) `heart`, NMI


(e) `austra`, Fscore

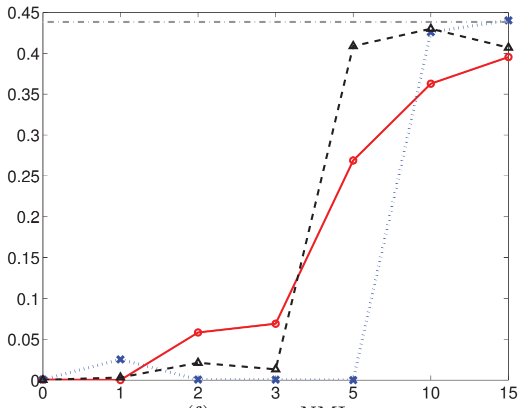
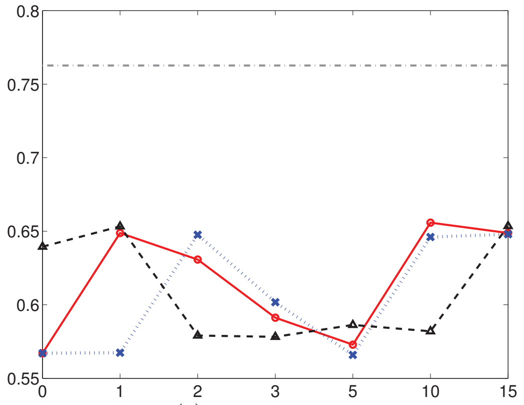
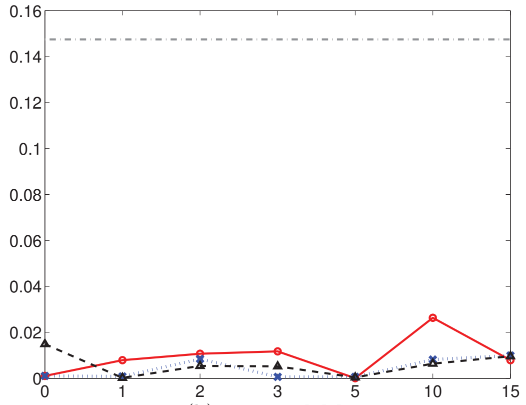

(f) `austra`, NMI

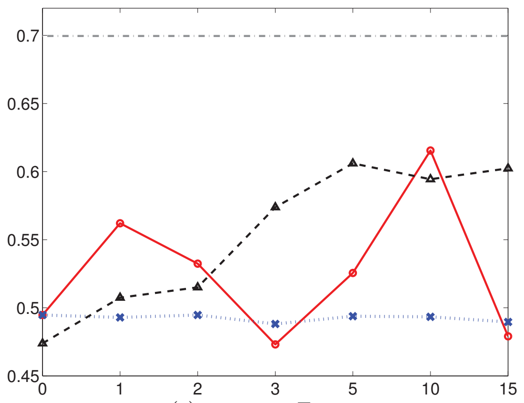
Figure 5.10: F-score and NMI for different algorithms for clustering under constraints for the data sets `UCI-seg`, `heart` and `austra`. The results of the proposed algorithm, `Shental`, and `Basu` are represented by the red solid line, blue dotted lines and the black dashed line, respectively. The performance of a classifier trained using all the labels is shown by the gray dashdot line. The horizontal axis shows the number of constraints as the percentage of the number of data points.



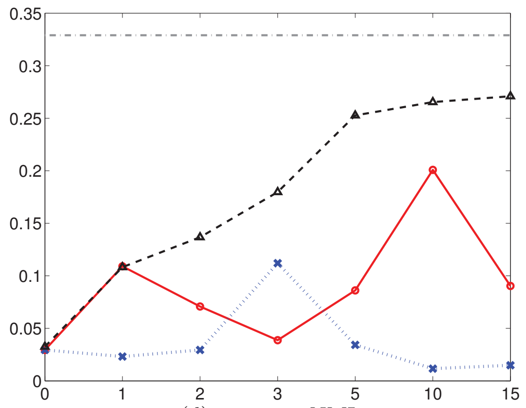
(a) `german`, Fscore



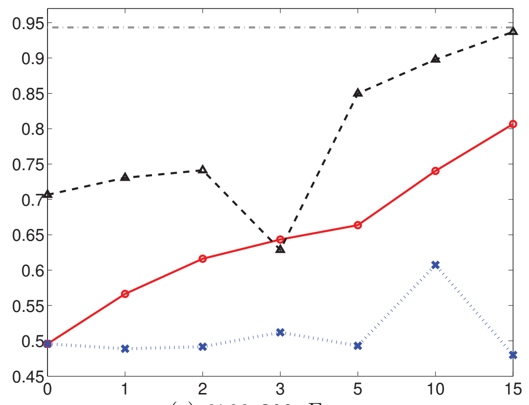
(b) `german`, NMI



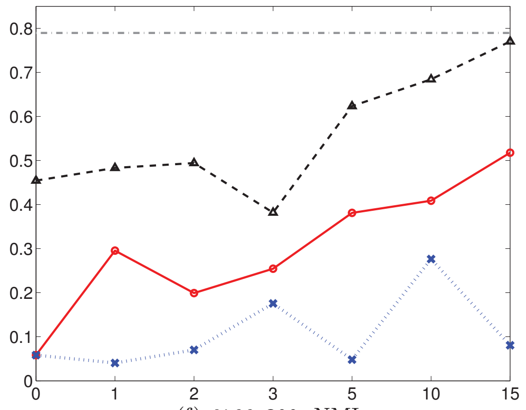
(c) `sim-300`, Fscore



(d) `sim-300`, NMI

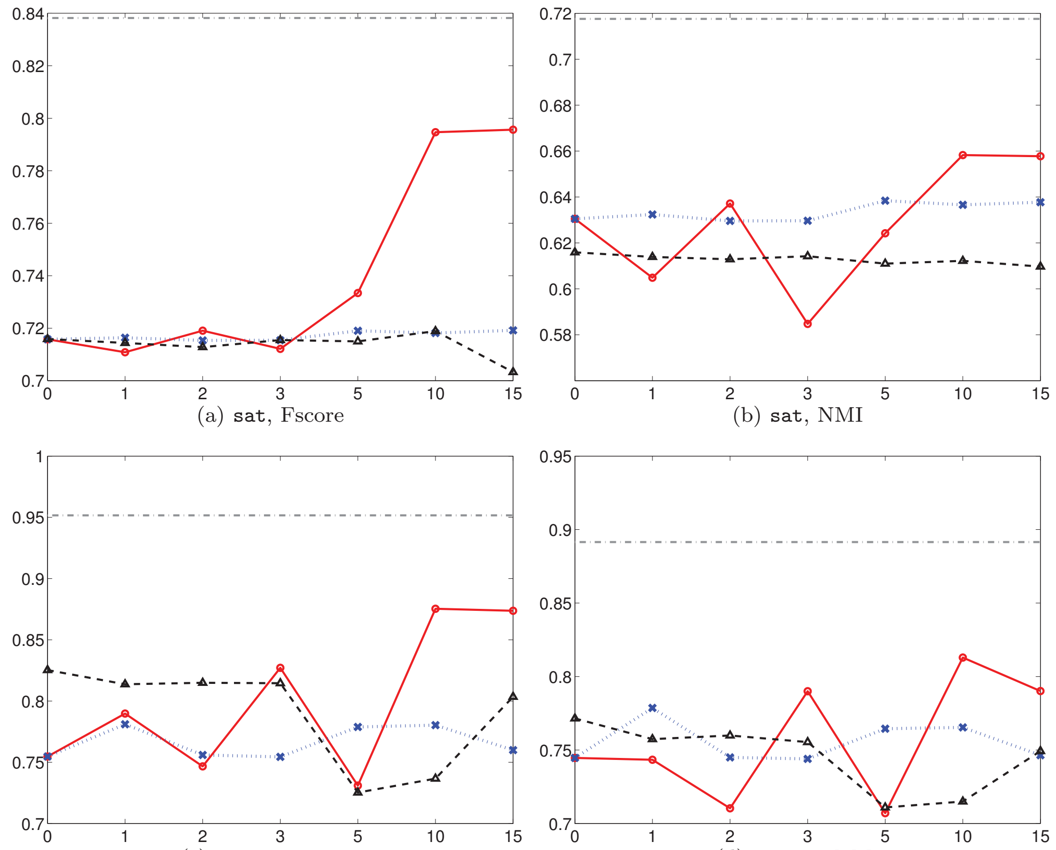


(e) `diff-300`, Fscore



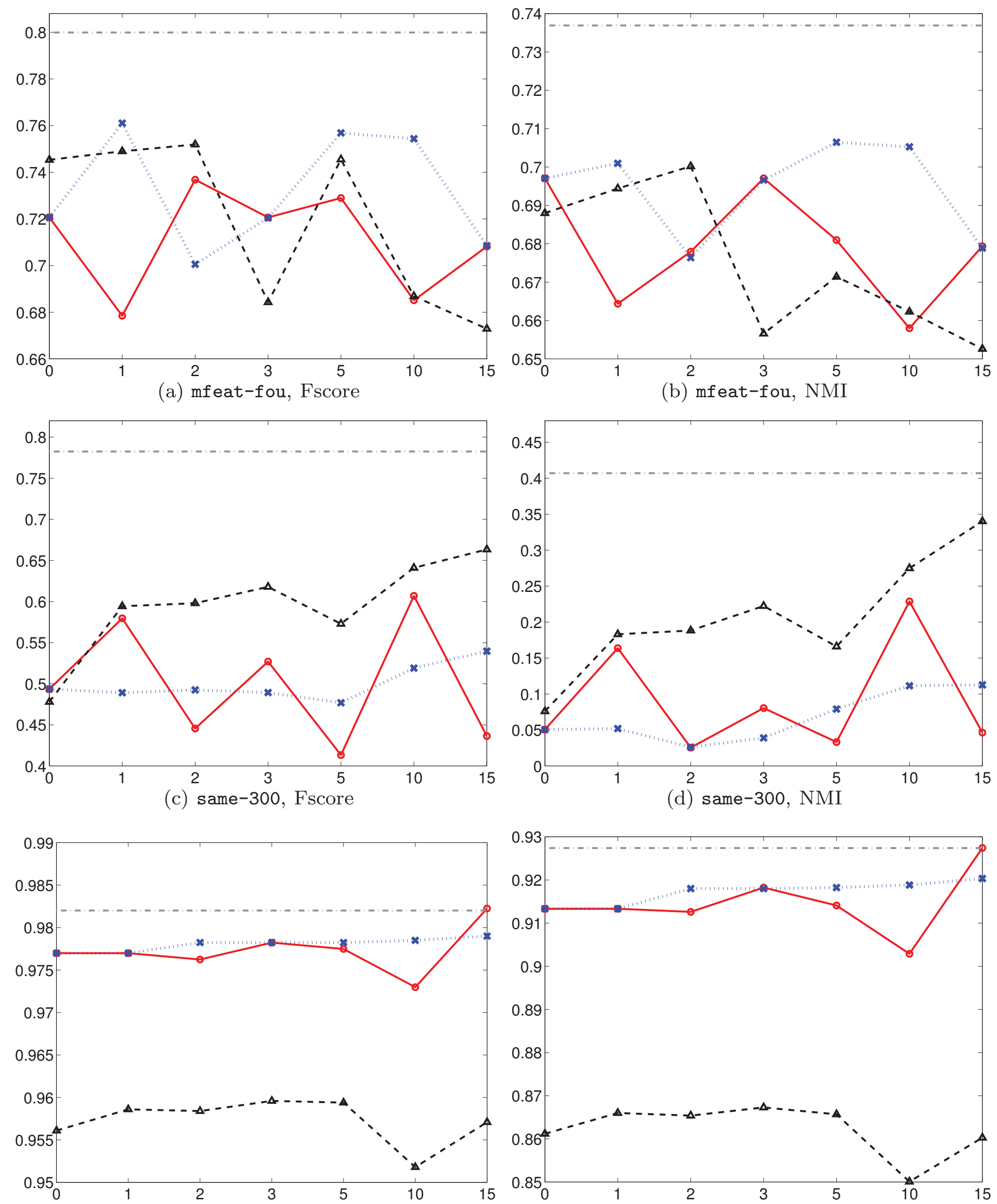
(f) `diff-300`, NMI

Figure 5.11: F-score and NMI for different algorithms for clustering under constraints for the data sets `german`, `sim-300` and `diff-300`. The results of the proposed algorithm, `Shental`, and `Basu` are represented by the red solid line, blue dotted lines and the black dashed line, respectively. The performance of a classifier trained using all the labels is shown by the gray dashdot line. The horizontal axis shows the number of constraints as the percentage of the number of data points.



(a) `sat`, Fscore (b) `sat`, NMI (c) `digits`, Fscore (d) `digits`, NMI

Figure 5.12: F-score and NMI for different algorithms for clustering under constraints for the data sets `sat` and `digits`. The results of the proposed algorithm, `Shental`, and `Basu` are represented by the red solid line, blue dotted lines and the black dashed line, respectively. The performance of a classifier trained using all the labels is shown by the gray dashdot line. The horizontal axis shows the number of constraints as the percentage of the number of data points.

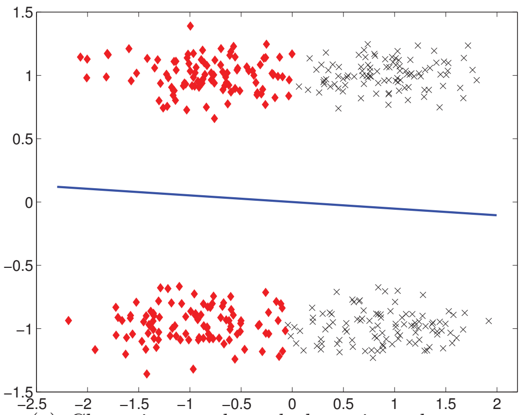


(a) `mfeat-fou`, Fscore (b) `mfeat-fou`, NMI

(c) `same-300`, Fscore (d) `same-300`, NMI

(e) `texture`, Fscore (f) `texture`, NMI

Figure 5.13: F-score and NMI for different algorithms for clustering under constraints for the data sets `mfeat-fou`, `same-300` and `texture`. The results of the proposed algorithm, `Shental`, and `Basu` are represented by the red solid line, blue dotted lines and the black dashed line, respectively. The performance of a classifier trained using all the labels is shown by the gray dashdot line. The horizontal axis shows the number of constraints as the percentage of the number of data points.


(a) Clustering result and the axis to be projected

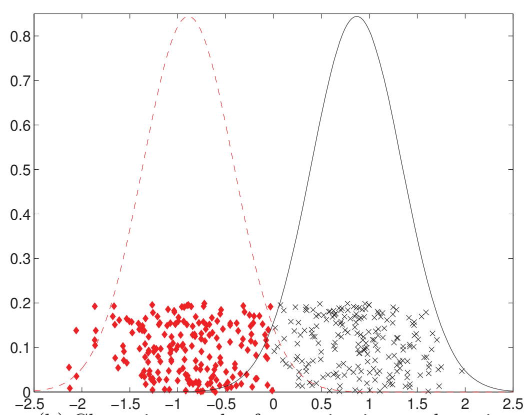
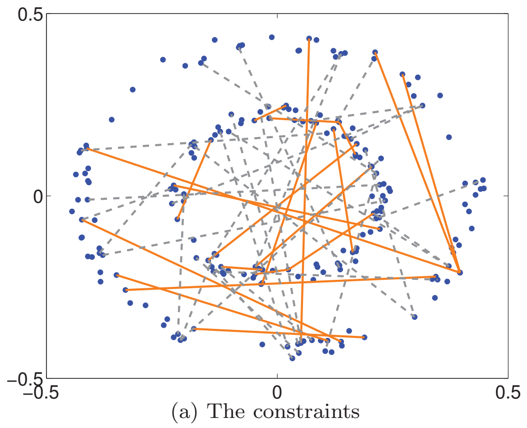
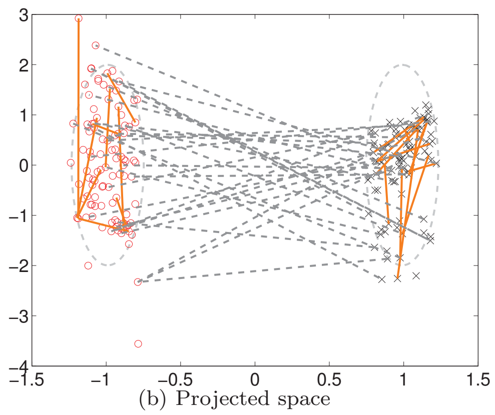
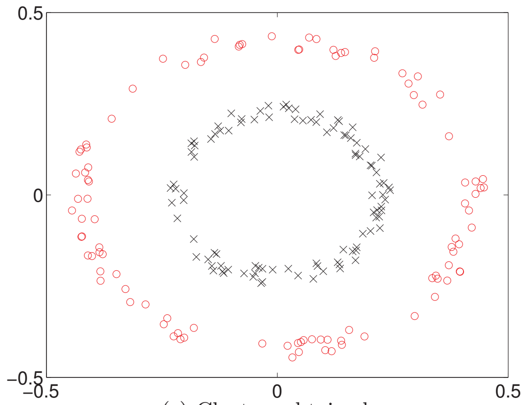

(b) Clustering result after projection to the axis

Figure 5.14: The result of simultaneously performing feature extraction and clustering with constraints simultaneously on the data set in Figure 5.3(a). The blue line in (a) corresponds to the projection direction found by the algorithm. The projected data points (which is 1D), together with the cluster labels and the two Gaussians, are shown in (b).


(a) The constraints


(b) Projected space


(c) Clusters obtained

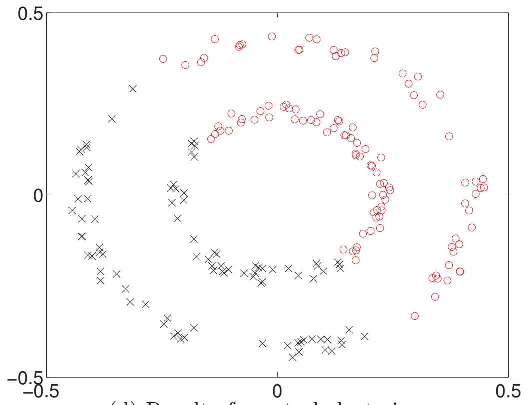

(d) Result of spectral clustering

Figure 5.15: An example of learning the subspace and the clusters simultaneously. (a): the original data and the constraints, where solid (dotted) lines correspond to must-link (must-not-link) constraints. (b) Clustering result of projecting 20 features extracted by kernel PCA to a 2D space. (c) Clustering solution (d) Result of applying spectral clustering [194] to this data set with two clusters, using the same kernel used for kernel PCA.

number of data points. Multiplication by the inverse of the Hessian in the line-search Newton algorithm can be performed in $O(d^3)$ time, because the structure of the $O(d^2)$ by $O(d^2)$ Hessian matrix is utilized for the inversion, and the matrix need not be formed explicitly. Unless the data set is very small, the time cost of inverting the Hessian is insignificant when compared with the calculation of the objective function $\mathcal{J}$ and its gradient. Therefore, one function evaluation of the proposed algorithm is only marginally slower than one iteration of the EM algorithm for the mixture model. Note that one Newton iteration can involve more than one function evaluation because of the line-search.

Each iteration in the algorithm `Shental` is similar to that in the standard EM algorithm. The difference is in the E-step, in which `Shental` involves an inference for a Markov network. This can take exponential time with respect to the number of constraints in the worst case. The per-iteration computation cost in `Basu` is in general smaller than both `Shental` and the proposed algorithm, because it is fundamentally the k-means algorithm. However, the use of iterative conditional mode to solve the cluster labels in the hidden Markov random fields, as well as the metric learning based on the constraints, becomes the overhead due to the constraints.

In practice, the proposed algorithm is slower than the other two because of the cross-validation procedure to determine the optimal λ. Even when λ is fixed, however, the proposed algorithm is still slower because (i) the optimization problem considered by the proposed algorithm is more difficult than those considered by `Shental` and `Basu`, and (ii) the convergence criteria based on the relative norm of gradient is stricter.

5.7.2 Discriminative versus Generative

One way to view the difference between the proposed algorithm and the algorithms `Shental` and `Basu` is that both `Shental` and `Basu` are generative, whereas the proposed approach is a combination of generative and discriminative. In supervised learning, a classifier is "generative" if it assumes a certain model on how the data from different classes are generated via the specification of the class conditional densities, whereas a "discriminative" classifier is built by optimizing some error measure, without any regard to the class conditional densities. Discriminative approaches are often superior to generative approaches when the actual class conditional densities differ from their assumed forms. On the other hand, incorporation of prior knowledge is easier for generative approaches because one can construct a generative model based on the domain knowledge. Discriminative approaches are also more prone to overfitting.

In the context of clustering under constraints, `Shental` and `Basu` can be regarded as generative because they specify a hidden Markov random field to describe how the data are generated. The constraint violation term $\mathcal{F}(\theta; \mathcal{C})$ used by the proposed algorithm is discriminative, because it effectively counts the number of violated constraints, which are analogous to the number of misclassified samples. The log-likelihood term $\mathcal{L}(\theta; \mathcal{Y})$ in the proposed objective function is generative because it is based on how the data are generated by a finite mixture model. Therefore, the proposed approach is both generative and discriminative, with the tradeoff parameter λ controlling the relative importance of these two properties. One can think that the discriminative component enables the proposed algorithm to have a higher performance, whereas the generative component acts as a regularization term to prevent overfitting in the discriminative component.

This discussion provides a new perspective in viewing the example in Figure 5.3. `Shental` and `Basu`, being generative, failed to recover the two desired clusters because their forms differ significantly from what `Shental` and `Basu` assume about a cluster. On the other hand, the discriminative property of the proposed algorithm can locate the desired vertical cluster boundary, which can satisfy the constraints.

The discriminative nature of the proposed algorithm is also the reason why the proposed algorithm, using constraints only, can outperform the generative classifier using all the labels. This is surprising at first, because, after all, constraints carry less information than labels. Incorporating the constraints on only some of the objects therefore should not outperform the case when the labels of all objects are available. However, this is only true when all possible classifiers are considered. When we restrict ourself to the generative classifier that assumes a Gaussian distribution with common covariance matrix as the class conditional density, it is possible for a discriminative algorithm to outperform the generative classifier if the class conditional densities are non-Gaussians. In fact, for the data sets `ethn`, `Mondrian`, `script`, `wdbc`, and `texture`, we observed that the proposed algorithm can have a higher F-score or NMI than that estimated using all the class labels. The difference is more noticeable for `script` and `wdbc`. Note that for the data set `austra`, the generative algorithm `Shental` can also out-perform the classifier trained using all the labels, though the difference is very small and it may be due to the noisy nature of this data set.

5.7.3 Drawback of the Proposed Approach

There are two main drawbacks of the proposed approach. The optimization problem considered, while accurately representing the goal of clustering with constraints, is more difficult. This has several consequences. First, a more sophisticated algorithm (line-search Newton) is needed instead of the simpler EM algorithm. The landscape of the proposed objective function is more "rugged". So, it it is more likely to get trapped in poor local optima. It also takes more iterations to reach a local optimum. Because we are initializing randomly, this also means that the proposed algorithm is not very stable if we have an insufficient number of random initializations.

The second difficulty is the determination of λ. (Note that the algorithm `Basu` has a similar parameter.) In our experiments, we adopted a cross-validation procedure to determine λ, which is computationally expensive. Cross-validation may yield a suboptimal λ when the number of informative constraints in the validation set is too small, or when too many constraints are erroneous due to the noise in the data. Here, a constraint is informative if it provides "useful" information to the clustering process. So, a must-link constraint between two points close to each other is not very informative because they are likely to be in the same cluster anyway.

Another problem is that we may encounter an unfavorable split of the training and validation constraints when the set of available constraints is too small. When this happens, the number of violations for the validation constraints is significantly larger than that of the training constraints. Increasing the value of λ cannot reduce the violation of the validation constraints, leading to an optimal constraint strength of zero. When this happens, we should try a different split of the constraints for training and validation.

5.7.4 Some Implementation Details

We have incorporated some heuristics in our optimization algorithm. During the optimization process, a cluster may become almost empty. This is detected when $\sum_i \tilde{r}_{ij}/n$ falls below a threshold, which is set to $4 \times 10^{-3}/k$. The empty cluster is removed, and the largest cluster that can result in the increase in the $\mathcal{J}$ value is split to maintain the same number of clusters. If no such cluster exists, the one that can lead to the smallest decrease in $\mathcal{J}$ is split. Another heuristic is that we lower-bound α_j by 10^{-8}, no matter what the values of $\{\beta_j\}$ are. This is used to improve the numerical stability of the proposed algorithm. The α_j are then renormalized to ensure that they sum to one.

5.8 Summary

We have presented an algorithm that handles instance-level constraints for model-based clustering. The key assumption in our approach is that the cluster labels are determined based on the feature vectors and the cluster parameters; the set of constraints has no influence here.. This contrasts with previous approaches like [231] and [21] which impose prior distribution on the cluster labels directly to reflect the constraints. This is the fundamental reason for the anomaly described in Section 5.2. The actual clustering is performed by the line-search Newton algorithm under the natural parameterization of the Gaussian distributions. The strength of the constraints is determined by a hold-out set of validation constraints. The proposed approach can be extended to handle simultaneously feature extraction and clustering under constraints. The effectiveness of the proposed approach has been demonstrated on both synthetic data sets and real-world data sets from different domain. In particular, we notice that the discriminative nature of the proposed algorithm can lead to superior performance when compared with a generative classifier trained using the labels of all the objects.

Chapter 6

Summary

The primary objective of the work presented in this dissertation is to advance the state-of-the-art in unsupervised learning. Unsupervised learning is challenging because its objective is often ill-defined. Instead of providing yet another new unsupervised learning algorithm, we are more interested in studying issues that are generic to different unsupervised learning tasks. This is the motivation behind the study of various topics in this dissertation, including the modification of the batch version of an algorithm to become incremental, the selection of the appropriate data representation (feature selection), and the incorporation of side-information in an unsupervised learning task.

6.1 Contributions

The results in this thesis have contributed to the field of unsupervised learning in several ways, and has led to the publication of two journal articles [163, 164]. Several conference papers [168, 161, 167, 165, 82] have also been published at different stages of the research conducted in this thesis.

The incremental ISOMAP algorithm described in Chapter 3 has made the following contributions:

- *Framework for incremental manifold learning:* The proposed incremental ISOMAP algorithm can serve as a general framework for converting a manifold learning algorithm to become incremental: the neighborhood graph is first updated, followed by the update of the low-dimensional representation, which is often an incremental eigenvalue problem similar to our case.

- *Solution of the all-pairs shortest path problems:* One component in the incremental algorithm is to update the all-pairs shortest path distances in view of the change in the neighborhood graph due to the new data points. We have developed a new algorithm that performs such an update efficiently. Our algorithm updates the shortest path distances from multiple source vertices simultaneously. This contrasts with previous work like [193], where different shortest path trees are updated independently.

- *Improved embedding for new data points:* We have derived an improved estimate of the inner product between the low-dimensional representation of the new point and the low-dimensional representations of the existing points. This leads to an improved embedding for the new point.

- *Algorithm for incremental eigen-decomposition with increasing matrix size:* The problem of updating the low-dimensional representation of the data points is essentially an incremental eigen-decomposition problem. Unlike the previous work [270], however, the size of the matrix we considered is increasing.

- *Vertex contraction to memorize the effect of data points:* A vertex contraction procedure that improves the geodesic distance estimate without additional memory is proposed.

Our work on estimating the feature saliency and the number of clusters simultaneously in Chapter 4 has made the following contributions:

- *Feature Saliency in unsupervised learning:* The problem of feature selection/feature saliency estimation is rarely studied for unsupervised learning. We tackle this problem by introducing a notion of feature saliency, which is able to describe the difference between the distributions of a feature among different clusters. The saliency is estimated efficiently by the EM algorithm.

- *Automatic Feature Saliency and Determination of the Number of Clusters:* The algorithm in [81], which utilizes the minimum message length to select the number of clusters automatically, is extended to estimate the feature saliency.

The clustering under constraints algorithm proposed in Chapter 5 has made the following contributions:

- *New objective function for clustering under constraints:* We have proposed a new objective function for clustering under constraints under the assumption that the constraints do not have any direct influence on the cluster labels. Extensive experimental evaluations reveal that this objective function is superior to the other state-of-the-art algorithms in most cases. It is also easy to extend the proposed objective function to handle group constraints that involve more than two data points.

- *Avoidance of Counter-intuitive Clustering Result:*

  The proposed objective function can avoid the pitfall of previous clustering under constraints algorithms like [231] and [21], which are based on hidden Markov random field. Specifically, clustering solutions that assign the cluster label to a data point that is different from all its neighbors is possible for previous algorithms, a situation avoided by the proposed algorithm.

- *Robustness to model-mismatch:*

  The proposed objective function for clustering under constraints is a combination of generative and discriminative terms. The discriminative term, which is based on the satisfaction of the constraints, improves the robustness of the proposed algorithm towards mismatch in the cluster shape. This leads to an improvement in the overall performance. The improvement can sometimes be so significant that the proposed algorithm, using constraints only, outperforms a generative supervised classifier trained using all the labels.

- *Feature extraction and clustering with constraints:* The proposed algorithm has been extended to perform feature extraction and clustering with constraints simultaneously by locating the best low-dimensional subspace, such that the Gaussian clusters formed will satisfy the given set of constraints as well as they can. This allows the proposed algorithm to handle data sets with higher dimensionality. The combination of this notion of feature extraction and the kernel trick allows us to extract clusters with general shapes.

- *Efficient implementation of the Line-search Newton Algorithm:*

The proposed objective function is optimized by the line-search Newton algorithm. The multiplication by the inverse of the Hessian for the case of a Gaussian mixture can be done efficiently with time complexity $O(d^3)$ without forming the $O(d^2)$ by $O(d^2)$ Hessian matrix explicitly. Here, d denotes the number of features. A naive approach of inverting the Hessian would require $O(d^6)$ time.

6.2 Future work

The study conducted in this dissertation leads to several interesting new research possibilities.

- *Improvement in the efficiency of the incremental ISOMAP algorithm*

  There are several possibilities for improving the efficiency of the proposed incremental ISOMAP algorithm. Data structures such as *kd*-tree, ball-tree, and cover-tree [19] can be used to speed up the search of the k nearest neighbors. The update strategy for geodesic distance and co-ordinates can be more aggressive; we can sacrifice the theoretical convergence property in favor of empirical efficiency. For example, the geodesic distance can be updated approximately using a scheme analogous to the distance vector protocol in the network routing literature. Co-ordinate update can be made faster if only a subset of the co-ordinates (such as those close to the new point) are updated at each iteration. The co-ordinates of every point would be finally updated if the new points came from different regions of the manifold.

- *Incrementalization of other manifold learning algorithms*

  The algorithm in Chapter 3 modifies the ISOMAP algorithm to become incremental. We can also modify similar algorithms, such as locally linear embedding or Laplacian eigenmap to become incremental.

- *Features dependency in dimensionality reduction and unsupervised learning:* The algorithm in Chapter 4 assumes that the features are conditionally independent of each other when the cluster labels are known. This assumption, however, is generally not true in practice. A new algorithm needs to be designed to cope with the situation when features are highly correlated in this setting.

- *Feature selection and constraints:*

  The main difficulty of feature selection in clustering is the ill-posed nature of the problem. A possible way to make the problem more well-defined is to introduce instance-level constraints. In Section 5.5, we described an algorithm for performing feature extraction and clustering under constraints simultaneously. One can apply a similar idea and use the constraints to assist in feature selection for clustering.

- *More efficient algorithms for clustering with constraints*

  The use of line-search Newton algorithm for optimizing the objective function in Chapter 5 is relatively efficient when compared with alternative approaches. Unfortunately, the objective function, which effectively uses Jensen-Shannon divergence to count the number of violated constraints, is difficult to optimize. It is similar to the minimization of the number of classification errors directly in supervised learning, which is generally perceived as difficult. Often, the number of errors is approximated by some quantities that are easier to optimize, such as the distances of mis-classified points from the separating hyperplane in the case of support vector machines. In the current context, we may want to approximate the number of violated constraints by some quantities that are easier to optimize. A difficulty can arise, however, when both must-link and must-not-link constraints are considered. If the violation of a must-link constraint is approximated by a convex function $g(.)$, the violation of a must-not-link constraint is naturally approximated by $-g(.)$, which is concave. Their combination leads to a function that is neither concave nor convex, which is difficult to optimize. Techniques like DC (difference of convex functions) programming [117] can be adopted for global optimization.

- *Number of clusters for clustering with constraints*

  The algorithm described in Chapter 5 assumes that the number of clusters is known. It is desirable if the number of clusters can be estimated automatically from the data. The presence of constraints should be helpful in this process. In fact, correlation clustering [10] considers must-link and must-not-link constraints only, without any regard to the feature vectors, and it can infer the optimal number of clusters by minimizing the number of constraint violations.

Appendices

Appendix A

Details of Incremental ISOMAP

In this appendix, we present the proof for the correctness of the algorithms in chapter 3 as well as analyzing their time complexity.

A.1 Update of Neighborhood Graph

The procedure to update the neighborhood graph has been described in section 3.2.1.1, where $\mathcal{A}$, the set of edges to be added, and $\mathcal{D}$, the set of edges to be deleted, are constructed upon insertion of v_{n+1} to the neighborhood graph.

Time Complexity For time complexity, note that for each i, the conditions in Equations (3.1) and (3.2) can be checked in constant time. So, the construction of $\mathcal{A}$ and $\mathcal{D}$ takes $O(n)$ time. The calculation of ι_i for all i can be done in $O(\sum_{i=1}^{n} deg(v_i)+|\mathcal{A}|)$ or $O(|E|+|\mathcal{A}|)$ time by examining the neighbors of different vertices. Here, $deg(v_i)$ denotes the degree of v_i. The complexity of the update of neighborhood graph can be bounded by $O(nq)$, where q is the maximum degree of the vertices in the graph after inserting v_{n+1}. Note that ι_i becomes the τ_i for the updated neighborhood graph.

A.2 Update of Geodesic Distances: Edge Deletion

A.2.1 Finding Vertex Pairs For Update

In this section, we examine how the geodesic distances should be updated upon edge deletion. Consider an edge $e(a, b) \in \mathcal{D}$ that is to be deleted. If $\pi_{ab} \neq a$, the shortest path between v_a and v_b does not contain $e(a, b)$. Deletion of $e(a, b)$ does not affect $sp(a, b)$ and hence none of the existing shortest paths are affected. Therefore, we have

Lemma A.1. *If $\pi_{ab} \neq a$, deletion of $e(a, b)$ does not affect any of the existing shortest paths and therefore no geodesic distance g_{ij} needs to be updated.*

We now consider the case $\pi_{ab} = a$. This implies $\pi_{ba} = b$ because the graph is undirected. The next lemma is an easy consequence of this assumption.

Lemma A.2. *For any vertex v_i, $sp(i, b)$ passes through v_a iff $sp(i, b)$ contains $e(a, b)$ iff $\pi_{ib} = a$.*

Before we proceed further, recall the definitions of $\mathcal{T}(b)$ and $\mathcal{T}(b; a)$ in section 3.1: $\mathcal{T}(b)$ is the shortest path tree of v_b, where the root node is v_b and $sp(b, j)$ consists of the tree edges from v_b to v_j, and $\mathcal{T}(b; a)$ is the subtree of $\mathcal{T}(b)$ rooted at v_a.

Let $R_{ab} \equiv \{i : \pi_{ib} = a\}$. Intuitively, R_{ab} contains vertices whose shortest paths to v_b include $e(a,b)$. We shall first construct R_{ab}, and then "propagate" from R_{ab} to get the geodesic distances that require update.

Because $sp(t,b)$ passes through the vertices that are the ancestor of v_t in $\mathcal{T}(b)$, plus v_t, we have

Lemma A.3. $R_{ab} = \{$ *vertices in* $\mathcal{T}(b;a)$ $\}$.

Proof.

$$\begin{aligned} & v_t \in \mathcal{T}(b;a) \\ \Leftrightarrow \quad & v_a \text{ is an ancestor of } v_t \text{ in } \mathcal{T}(b), \text{ or } v_a = v_t \\ \Leftrightarrow \quad & sp(t,b) \text{ passes through } v_a \\ \Leftrightarrow \quad & \pi_{tb} = a \qquad \text{(lemma A.2)} \\ \Leftrightarrow \quad & t \in R_{ab} \end{aligned}$$

□

If v_t is a child of v_u in $\mathcal{T}(b)$, v_u is the vertex in $sp(b,t)$ just before v_t. Thus, we have the lemma below.

Lemma A.4. *The set of children of* v_u *in* $\mathcal{T}(b) = \{v_t : v_t$ *is a neighbor of* v_u *and* $\pi_{bt} = u\}$.

Consequently, we can examine all the neighbors of v_u to find the node's children in $\mathcal{T}(b)$ based on the predecessor matrix. Note that the shortest path trees are not stored explicitly; only the predecessor matrix is maintained. The first nine lines in Algorithm 3.1 perform a tree traversal that extracts all the vertices in $\mathcal{T}(b;a)$ to form R_{ab}, using Lemma A.4 to find all the children of a node in the tree.

Time Complexity At any time, the queue Q contains vertices in the subtree $\mathcal{T}(b;a)$ that have been examined. The while-loop is executed $|R_{ab}|$ times because a new vertex is added to R_{ab} in each iteration. The inner for-loop is executed a total of $\sum_{v_t \in R_{ab}} deg(v_t)$, which can be bounded loosely by $q|R_{ab}|$. Therefore, a loose bound for the first nine lines in Algorithm 3.1 is $O(q|R_{ab}|)$.

A.2.2 Propagation Step

Define $F_{(a,b)} \equiv \{(i,j) : sp(i,j)$ contains $e(a,b)\}$. Here, (a,b) denotes the unordered pair a and b. So, $F_{(a,b)}$ is indexed by the unordered pair (a,b), and its elements are also unordered pairs. Intuitively, $F_{(a,b)}$ contains the vertex pairs whose geodesic distances need to be recomputed when the edge $e(a,b)$ is deleted. Starting from v_b for each of the vertex in R_{ab}, we construct $F_{(a,b)}$ by a search.

Lemma A.5. *If* $(i,j) \in F_{(a,b)}$, *either* i *or* j *is in* R_{ab}.

Proof. $(i,j) \in F_{(a,b)}$ is equivalent to $sp(i,j)$ contains $e(a,b)$. The shortest path $sp(i,j)$ can be written either as $sp(i,j) = v_i \rightsquigarrow v_a \rightarrow v_b \rightsquigarrow v_j$, or $sp(i,j) = v_i \rightsquigarrow v_b \rightarrow v_a \rightsquigarrow v_j$, where $\rightsquigarrow$ denotes a path between the two vertices. Because the subpath of a shortest path is also a shortest path, either $sp(i,b)$ or $sp(j,b)$ passes through v_a. By lemma A.2, either $\pi_{ib} = a$ or $\pi_{jb} = a$. Hence either i or j is in R_{ab}. □

Lemma A.6. $F_{(a,b)} = \bigcup_{u \in R_{ab}} \{(u,t) : v_t$ *is in* $\mathcal{T}(u;b)\}$.

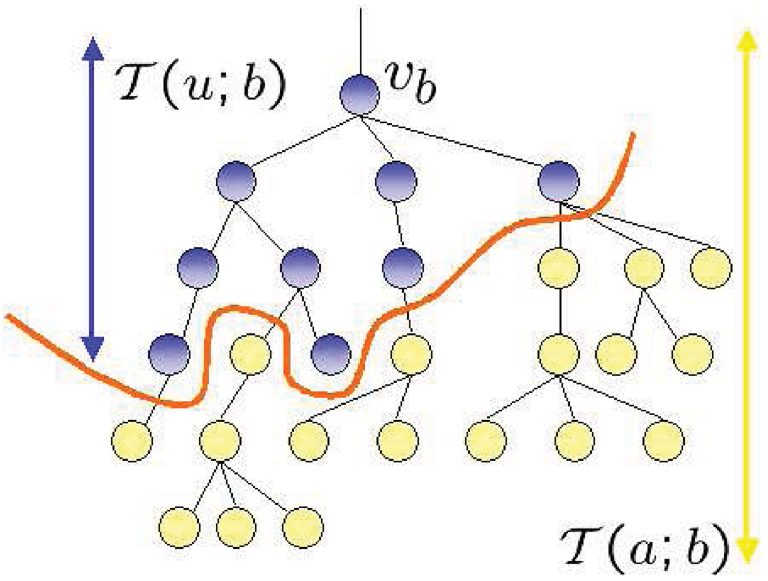


Figure A.1: Example of $\mathcal{T}(u;b)$ and $\mathcal{T}(a;b)$. All the nodes and the edges shown constitute $\mathcal{T}(a;b)$, whereas only the part of the subtree above the line constitutes $\mathcal{T}(u;b)$. This example illustrates the relationship of $\mathcal{T}(u;b)$ and $\mathcal{T}(a;b)$ as proved in Lemma A.7.

Proof. By lemma A.5, $(u,t) \in F_{(a,b)}$ implies either u or t is in R_{ab}. Without loss of generality, suppose $u \in R_{ab}$. So, $sp(u,t)$ can be written as $v_u \rightsquigarrow v_a \to v_b \rightsquigarrow v_t$. Thus v_t must be in $\mathcal{T}(u;b)$. On the other hand, for any vertex v_t in the subtree of $\mathcal{T}(u;b)$, $sp(u,t)$ goes through v_b. Since $sp(u,b)$ goes through v_a (because $u \in R_{ab}$), $sp(u,t)$ must also go through v_a and hence use $e(a,b)$. □

Direct application of the above lemma to compute $F_{(a,b)}$ requires the construction of $\mathcal{T}(u;b)$ for different u. This is not necessary, however, because for all $u \in R_{ab}$, $\mathcal{T}(u;b)$ must be a part of $\mathcal{T}(a;b)$ in the sense that is exemplified in Figure A.1. This relationship aids the construction of $\mathcal{T}(u;b)$ in Algorithm 3.1 (the variable $\mathcal{T}'$) because we only need to expand the vertices in $\mathcal{T}(a;b)$ that are also in $\mathcal{T}(u;b)$.

Lemma A.7. *Consider $u \in R_{ab}$. The subtree $\mathcal{T}(u;b)$ is non-empty, and let v_t be any vertex in this subtree. Let v_s be a child of v_t in $\mathcal{T}(u;b)$, if any. We have the following:*

1. *v_t is in the subtree of $\mathcal{T}(a;b)$.*
2. *v_s is a child of v_t in the subtree of $\mathcal{T}(a;b)$.*
3. *$\pi_{us} = \pi_{as} = t$*

Proof. The subtree $\mathcal{T}(u;b)$ is not empty because v_b is in this subtree. For any v_t in this subtree, $sp(u,t)$ passes through v_b. Hence $sp(u,b)$ is a subpath of $sp(u,t)$. Because $u \in R_{ab}$, $sp(u,b)$ passes through v_a. So, we can write $sp(u,t)$ as $v_u \rightsquigarrow v_a \to v_b \rightsquigarrow v_t$. So, $sp(a,t)$ contains v_b, and this implies that v_t is in $\mathcal{T}(a;b)$.

Now, if v_s is a child of v_t in $\mathcal{T}(u;b)$, $sp(u,s)$ can be written as $v_u \rightsquigarrow v_a \to v_b \rightsquigarrow v_t \to v_s$. So, $\pi_{us} = t$. Because any subpath of a shortest path is also a shortest path, $sp(a,s)$ is simply $v_a \to v_b \rightsquigarrow v_t \to v_s$, which implies that v_s is also a child of v_t in $\mathcal{T}(a;b)$, and $\pi_{as} = t$. Therefore, we have $\pi_{us} = \pi_{as} = t$. □

Let F be the set of unordered pair (i, j) such that a new shortest path from v_i to v_j is needed when edges in $\mathcal{D}$ are removed. So, $F = \bigcup_{e(a,b)\in\mathcal{D}} F_{(a,b)}$. For each $(a, b) \in \mathcal{D}$, R_{ab} constructed in the first nine lines in Algorithm 3.1 is used to construct $F_{(a,b)}$ from line 11 until the end of Algorithm 3.1. At each iteration of the while-loop starting at line 15, the subtree $\mathcal{T}(a; b)$ is traversed, using the condition $\pi_{us} = \pi_{as}$ to check if v_s is in $\mathcal{T}(u; b)$ or not. The part of the subtree $\mathcal{T}(a; b)$ is expanded only when necessary, using the variable $\mathcal{T}'$.

Time Complexity If we ignore the time to construct $\mathcal{T}'$, the complexity of the construction of F is proportional to the number of vertices examined. If the maximum degree of $\mathcal{T}'$ is q', this is bounded by $O(q'|F|)$. Note that $q' \leq q$, where q is the maximum degree of the vertices in the neighborhood graph. The time to expand $\mathcal{T}'$ is proportional to the number of vertices actually expanded plus the number of edges incident on those vertices. This is bounded by q times the size of the tree, and the size of the tree is at most $O(|F_{(a,b)}|)$. Usually, the time is much less, because different u in R_{ab} can reuse the same $\mathcal{T}'$. The time complexity to construct $F_{(a,b)}$ can be bounded by $O(q|F_{(a,b)}|)$ in the worst case. The overall time complexity to construct F, which is the union of $F_{(a,b)}$ for all $(a, b) \in \mathcal{D}$, is $O(q|F|)$, assuming the number of duplicate pairs in $F_{(a,b)}$ for different (a, b) is $O(1)$. Empirically, there are at most several such pairs. Most of the time, there is no duplicate pair at all.

A.2.3 Performing The Update

Let $\mathcal{G}' = (V, E/\mathcal{D})$, the graph after deleting the edges in $\mathcal{D}$. Let $\mathcal{B}$ be an auxiliary undirected graph with the same vertices as $\mathcal{G}$, but its edges are based on F. In other words, there is an edge between v_i and v_j in the graph $\mathcal{B}$ if and only if (i, j) is in F. Because F contains all the vertex pairs whose geodesic distances need to be updated, an edge in $\mathcal{B}$ corresponds to a geodesic distance value that needs to be revised.

To update the geodesic distances, we first pick a v_u in $\mathcal{B}$ with at least one edge incident on it. Define $C(u) = \{i : e(u, i) \text{ is an edge of } \mathcal{B}\}$. So, the geodesic distance $g_{u,i}$ needs to be updated if and only if $i \in C(u)$. These geodesic distances are updated by the modified Dijkstra's algorithm (Algorithm 3.2), with v_u as the source vertex and $C(u)$ as the set of "unprocess vertices", i.e., the set of vertices such that their shortest paths from v_u are invalid. Recall the basic idea of Dijkstra's algorithm is that, starting with an empty set of "processed vertices" (vertices whose shortest paths have been found), different vertices are added one by one to this set in an ascending order of estimated shortest path distances. The ascending order guarantees the optimality of the shortest paths. Algorithm 3.2 does something similar, except that the set of "processed vertices" begins with $V/C(u)$ instead of an empty set. The first for-loop estimates the shortest path distances for $j \in C(u)$ if $sp(u, j)$ is "one edge away" from the processed vertices, i.e., $sp(u, j)$ can be written as $v_u \rightsquigarrow v_a \to v_j$ with $a \in V/C(u)$. In the while loop, the vertex v_k $(k \in C(u))$ with the smallest estimated shortest path distance is examined and transferred into the set of processed vertices. The estimates of the shortest path distances between v_u and the adjacent vertices of v_k are relaxed (updated) accordingly. This repeats until $C(u)$ becomes empty, i.e., all vertices have been processed.

When the modified Dijkstra's algorithm with v_u as the source vertex finishes, all geodesic distances involving v_u have been updated. Since an edge in $\mathcal{B}$ corresponds to a geodesic distance estimate requiring update, we should remove all edges incident on v_u in $\mathcal{B}$. We then select another vertex $v_{u'}$ with at least one edge incident on it in $\mathcal{B}$, and call the modified Dijkstra's algorithm again but with $v_{u'}$ as the source vertex. This repeats until $\mathcal{B}$ becomes an empty graph.

Time Complexity The for-loop in Algorithm 3.2 takes at most $O(q|C(u)|)$ time. In the while-loop, there are $|C(u)|$ ExtractMin operations, and the number of DecreaseKey operations depends on how many edges are there within the vertices in $C(u)$. A upper bound for this is $q|C(u)|$. By using Fibonacci's heap, ExtractMin can be done in $O(\log |C(u)|)$ time while DecreaseKey can be done in $O(1)$ time, on average. Thus the complexity of algorithm 3.2 is $O(|C(u)| \log |C(u)| + q|C(u)|)$. If binary heap is used instead, the complexity is $O(q|C(u)| \log |C(u)|)$.

A.2.4 Order for Performing Update

How do we select v_u in $\mathcal{B}$ to be eliminated and to act as the source vertex for the modified Dijkstra's Algorithm (Algorithm 3.2)? We seek an elimination order that minimizes the time complexity of all the updates. Let f_i be the degree of v_{κ_i}, the i-th vertex removed from $\mathcal{B}$. So, $f_i = |C(\kappa_i)|$. The overall time complexity T for running the modified Dijkstra's algorithm (with Fibonacci's heap) for all the vertices in $\mathcal{B}$ with at least an incident edge is $O(T)$, with

$$T = \sum_i (f_i \log f_i + q f_i). \tag{A.1}$$

Because $\sum_{i=1}^n f_i$ is a constant (twice the number of edges in $\mathcal{B}$) with respect to different elimination order, the vertices should be eliminated in an order that minimizes $\sum_i f_i \log f_i$. If binary heap is used, the time complexity is $O(T^*)$, with

$$T^* = q \sum_i f_i \log f_i. \tag{A.2}$$

In both cases, we should minimize $\sum_i f_i \log f_i$. Finding an order that minimizes this is difficult, unfortunately. Since this sum is dominated by the largest f_i, we instead minimize $\max_i f_i$. This minimization is achieved by a greedy algorithm that removes the vertex in $\mathcal{B}$ with the smallest degree. The correctness of this greedy approach can be seen from the following argument. Suppose the greedy algorithm is wrong. So, at some point the algorithm makes a mistake, i.e., the removal of v_t instead of v_u leads to an increase of $\max_i f_i$. This can only happen when $deg(v_t) > deg(v_u)$. We get a contradiction, since the algorithm always removes the vertex with the smallest degree.

Because the degree of each vertex is an integer, an array of linked lists can be used to implement the greedy search (Algorithm 3.3) efficiently without an explicit search. At any time of the instance, the linked list $l[i]$ is empty for $i < pos$. So, the vertex in $l[i]$ has the smallest degree in $\mathcal{B}$. The for-loop in lines 10 to 18 removes all the edges incident on v_j in $\mathcal{B}$ by reducing the degree of all vertices adjacent to v_j by one, and moving *pos* back by one if necessary.

Time Complexity The first for-loop in Algorithm 3.3 takes $O(|F|)$ time, because $|F|$ is the number of edges in $\mathcal{B}$. In the second for-loop, *pos* is incremented at most $2n$ times, because it can move backwards at most n steps. The inner for-loop is executed altogether $O(|F|)$ time. Therefore, the overall time complexity for algorithm 3.3 (excluding the time for executing the modified Dijkstra's algorithm) is $O(|F|)$.

A.3 Update of Geodesic Distances: Edge Insertion

In Equation (3.3), we describe how the geodesic distance between the new vertex v_{n+1} and v_i is computed, after updating the geodesic distance in view of the edge deletion. Since all the edges in

$\mathcal{A}$, the set of edges inserted into the neighborhood graph, are incident on v_{n+1}, any improvement in an existing shortest path must involve v_{n+1}. Let $L = \{(i, j) : w_{i,n+1} + w_{n+1,j} < g_{ij}\}$. Intuitively, L is the set of unordered pairs adjacent to v_{n+1} with improved shortest paths due to the insertion of v_{n+1}.

For different $(a, b) \in L$, Algorithm 3.4 is used to propagate the effect of the improvement in $sp(a, b)$ to the vertices near v_a and v_b. First, lines 1 to 9 construct a set S_{ab} that is similar to R_{ab} in Algorithm 3.1, and it consists of vertices whose shortest paths to v_b have been improved. For each vertex v_i in S_{ab}, lines 11 to 22 search for other shortest paths starting from v_i that can be improved, and update the geodesic distance according to the improved shortest path just discovered. Its idea is analogous to the construction of $F_{(a,b)}$ in Algorithm 3.1, but now $sp(a, b)$ is improved instead of destroyed as in the case of $F_{(a,b)}$.

The correctness of Algorithm 3.1 can be seen by the following argument. Without loss of generality, the improved shortest path between v_i and v_j can be written as $v_i \rightsquigarrow v_a \rightarrow v_{n+1} \rightarrow v_b \rightsquigarrow v_j$. So, v_i is a vertex in $\mathcal{T}(n + 1; a)$, and v_j must be in both $\mathcal{T}(i; b)$ and $\mathcal{T}(n + 1; b)$. If v_l is a child of v_j in $\mathcal{T}(i; b)$, v_l is also a child of v_j in $\mathcal{T}(n + 1; b)$, and $(g_{i,n+1} + g_{n+1,l}) < g_{il}$ should be satisfied. In other words, the relationship between $\mathcal{T}(i; b)$ and $\mathcal{T}(n + 1; b)$ here is similar to the relationship between $\mathcal{T}(u; b)$ and $\mathcal{T}(a; b)$ depicted in Figure A.1. The proof of these properties is similar to the proof given for the relationship between $F_{(a,b)}$ and R_{ab}, and hence is not repeated.

Time Complexity The set L can be constructed in $O(|\mathcal{A}|^2)$ time. Let $H = \{(i, j)$: A better shortest path appears between v_i and v_j because of v_{n+1} $\}$. By an argument similar to the complexity of constructing F, the complexity of finding H and revising the corresponding geodesic distances in Algorithm 3.4 is $O(q|H| + |\mathcal{A}|^2)$.

A.4 Geodesic Distance Update: Overall Time Complexity

Updating the neighborhood graph takes $O(nq)$ time. The construction of R_{ab} and F_{ab} (Algorithm 3.1) takes $O(q|R_{ab}|)$ and $O(q|F_{ab}|)$ time, respectively. Since $|F_{ab}| \geq |R_{ab}|$, these steps take $O(q|F_{ab}|)$ time together. As a result, F can be constructed in $O(q|F|)$ time. The time to run the modified Dijkstra's algorithm (Algorithm 3.2) is difficult to estimate. Let μ be the number of vertices in $\mathcal{B}$ with at least one edge incident on it, and let $\nu \equiv \max_i f_i$ with f_i defined in Appendix A.2.4. In the highly unlikely worst case, ν can be as large as μ. The time of running Algorithm 3.2 can be rewritten as $O(\mu\nu \log \nu + q|F|)$. The typical value of ν can be estimated using concepts from random graph theory. It is easy to see that

$$\nu = \max_l \{\mathcal{B} \text{ has a } l\text{-regular sub-graph}\}, \tag{A.3}$$

where a l-regular sub-graph is defined as a subgraph with the degree of all vertices as l. Unfortunately, we fail to locate the exact result on the behavior of the largest l-regular sub-graph in random graph theory. On the other hand, the largest l-complete sub-graph, i.e., a clique of size l, of a random graph has been well studied. The clique number (the size of the largest clique in a graph) of almost every graph is "close" to $O(\log \mu)$ [200], assuming the average degree of vertices is a constant and μ is the number of vertices in the graph. Based on our empirical observations in the experiments, we conjecture that, on average, ν is also of the order $O(\log \mu)$. With this conjecture, the total time to run the Dijkstra's algorithm can be bounded by $O(\mu \log \mu \log \log \mu + q|F|)$. Finally, the time complexity of algorithm 3.4 is $O(q|H| + |\mathcal{A}|^2)$. So, the overall time complexity can be written as $O(q|F| + q|H| + \mu \log \mu \log \log \mu + |\mathcal{A}|^2)$. Note that $\mu \leq 2|F|$. In practice, the first two terms dominate, and the complexity can be written as $O(q(|F| + |H|))$.

Appendix B

Calculations for Clustering with Constraints

The purpose of this appendix is to derive the results in Chapter 5, some of which are relatively involved.

B.1 First Order Information

In this appendix, we shall derive the gradient of the objective function $\mathcal{J}$. The differential of a variable or a function x will be denoted by "$d\ x$". We shall first compute the differential of $\mathcal{J}$, followed by the conversion of the differentials into the derivatives with respect to the cluster parameters.

B.1.1 Computing the Differential

The differential of the log-likelihood can be derived as follows:

$$\begin{aligned} d\ \mathcal{L}(\theta;\mathcal{Y}) &= \sum_{i=1}^{n} d\left(\log \sum_{j=1}^{k} \exp(\log q_{ij})\right) = \sum_{i=1}^{n}\sum_{j=1}^{k} \frac{\exp(\log q_{ij})\left(d\ \log q_{ij}\right)}{\sum_{j'} \exp(\log q_{ij'})} \\ &= \sum_{i=1}^{n}\sum_{j=1}^{k} r_{ij}\left(d\ \log q_{ij}\right). \end{aligned} \tag{B.1}$$

Here, $r_{ij} = \exp(\log q_{ij})/\sum_{j^*}\exp(\log q_{ij^*}) = q_{ij}/\sum_{j^*} q_{ij^*}$ is the usual posterior probability for the j-th cluster given the point $\mathbf{y}_i$. The annealing version of the log-likelihood, which is needed if we want to apply a deterministic-annealing type of procedure to optimize the log-likelihood, is defined by

$$\mathcal{L}^{\text{annealed}}(\theta;\mathcal{Y},\gamma) = \sum_{i=1}^{n}\sum_{j=1}^{k} \tilde{r}_{ij}\log q_{ij} - \frac{1}{\gamma}\sum_{i=1}^{n}\sum_{j=1}^{k} \tilde{r}_{ij}\log \tilde{r}_{ij}, \tag{B.2}$$

where γ is the inverse temperature parameter. Note that $\mathcal{L}^{\text{annealed}}(\theta;\mathcal{Y},\gamma)$ becomes the log-likelihood $\mathcal{L}(\theta;\mathcal{Y})$ when γ is one. The temperature *invtemp* is different from the smoothness parameter τ: γ is related to all the data points, whereas τ is only concerned with objects involved in the constraints. The "fuzzy" cluster assignment $\tilde{r}_{ij}$ is defined as

$$\tilde{r}_{ij} = \frac{q_{ij}^{\gamma}}{\sum_{j'} q_{ij'}^{\gamma}}. \tag{B.3}$$

A small value of γ corresponds to a state of high temperature, in which the cluster assignments are highly uncertain. The first term in Equation (B.2) can also be understood as a weighted sum of distortion in coding theory, with $\tilde{r}_{ij}$ as the weights and $\log q_{ij}$ as the distortion. The second term in Equation (B.2) is proportional to the sum of the entropy of $\tilde{r}_{ij}$. Because

$$\begin{aligned}\mathcal{L}^{\text{annealed}}(\theta; \mathcal{Y}, \gamma) &= \sum_{ij} \tilde{r}_{ij} \log q_{ij} - \frac{1}{\gamma} \sum_{ij} \tilde{r}_{ij} \log q_{ij}^{\gamma} + \frac{1}{\gamma} \sum_{ij} \tilde{r}_{ij} \log \sum_{l} q_{il}^{\gamma} \\ &= \frac{1}{\gamma} \sum_{i} \log \sum_{l} \exp\left(\gamma \log q_{il}\right),\end{aligned}$$

the differential of the annealed log-likelihood is similar to that of the log-likelihood, which is

$$d\ \mathcal{L}^{\text{annealed}}(\theta; \mathcal{Y}, \gamma) = \sum_{ij} \tilde{r}_{ij} \left(d\ \log q_{ij}\right) \tag{B.4}$$

Our next step is to derive the differential for the constraint satisfaction function $\mathcal{F}(\theta; \mathcal{C})$. Based on the definition of s_{ij} in Equation (5.12), we can obtain its differential as

$$\begin{aligned}d\ \log s_{ij} &= d\ \left(\tau \log q_{ij}\right) - d\ \log \sum_{l} \exp\left(\tau \log q_{il}\right) \\ &= \tau \left(d\ \log q_{ij}\right) - \tau \sum_{l} s_{il} \left(d\ \log q_{il}\right) \\ d\ s_{ij} &= \tau s_{ij} \left(d\ \log q_{ij} - \sum_{j} s_{ij} \left(d\ \log q_{ij}\right)\right) \\ d\ t_{hj}^{+} &= \sum_{i} a_{hi} \left(d\ s_{ij}\right) \\ d\ t_{hj}^{-} &= \sum_{i} b_{hi} \left(d\ s_{ij}\right)\end{aligned}$$

Note that $\sum_{j=1}^{k} d\ s_{ij} = \sum_{j=1}^{k} d\ t_{hj}^{+} = \sum_{j=1}^{k} d\ t_{hj}^{-} = 0$ because $\sum_{j} s_{ij} = \sum_{j=1}^{k} t_{hj}^{+} = \sum_{j=1}^{k} t_{hj}^{-} = 1$. The differential for the negative entropy of s_{ij}, t_{hj}^{+} and t_{hj}^{-} can be derived by considering

$$\begin{aligned}
d\ \sum_j s_{ij}\log s_{ij} &= \sum_j (\log s_{ij}+1)(d\ s_{ij}) = \sum_j \log s_{ij}(d\ s_{ij}) \\
&= \tau \sum_j s_{ij}\log s_{ij}\left(d\ \log q_{ij} - \sum_l s_{il}\left(d\ \log q_{il}\right)\right) \\
&= \tau \sum_j \left(s_{ij}\log s_{ij} - s_{ij}\sum_l s_{il}\log s_{il}\right)(d\ \log q_{ij}) \\
d\ \sum_j t^+_{hj}\log t^+_{hj} &= \sum_j \log t^+_{hj}(d\ t^+_{hj}) \\
&= \tau \sum_j \log t^+_{hj}\sum_i a_{hi}s_{ij}\left(d\ \log q_{ij} - \sum_l s_{il}\left(d\ \log q_{il}\right)\right) \\
&= \tau \sum_{ij} a_{hi}s_{ij}\left(\log t^+_{hj} - \sum_l s_{il}\log t^+_{hl}\right)\left(d\ \log q_{ij}\right) \\
d\ \sum_j t^-_{hj}\log t^-_{hj} &= \tau \sum_{ij} b_{hi}s_{ij}\left(\log t^-_{hj} - \sum_l s_{il}\log t^-_{hl}\right)\left(d\ \log q_{ij}\right)
\end{aligned}$$

The differential for the Jensen-Shannon divergence term is then given by

$$\begin{aligned}
d\ D^+_{JS}(h) &= d\ \left(\sum_i a_{hi}\sum_j s_{ij}\log s_{ij} - \sum_j t^+_{hj}\log t^+_{hj}\right) \\
&= \tau \sum_{ij} a_{hi}s_{ij}\left(\log s_{ij} - \sum_l s_{il}\log s_{il}\right)\left(d\ \log q_{ij}\right) \\
&\qquad - \tau \sum_{ij} a_{hi}s_{ij}\left(\log t^+_{hj} - \sum_l s_{il}\log t^+_{hl}\right)\left(d\ \log q_{ij}\right) \\
&= \tau \sum_{ij} a_{hi}s_{ij}\left(\log \frac{s_{ij}}{t^+_{hj}} - \sum_l s_{il}\log \frac{s_{il}}{t^+_{hl}}\right)\left(d\ \log q_{ij}\right) \\
d\ D^-_{JS}(h) &= \tau \sum_{ij} b_{hi}s_{ij}\left(\log \frac{s_{ij}}{t^-_{hj}} - \sum_l s_{il}\log \frac{s_{il}}{t^-_{hl}}\right)\left(d\ \log q_{ij}\right)
\end{aligned}$$

The differential of the loss functions of constraint violation can thus be written as

$$\begin{aligned}
d\ \mathcal{F}(\theta;\mathcal{C}) &= d\left(-\sum_{h=1}^{m^+}\lambda_h^+ D_{JS}^+(h) + \sum_{h=1}^{m^-}\lambda_h^- D_{JS}^-(h)\right)\\
&= -\tau\sum_{ij}\Bigg(\sum_{h=1}^{m^+}\lambda_h^+ a_{hi}s_{ij}\Big(\log\frac{s_{ij}}{t_{hj}^+} - \sum_l s_{il}\log\frac{s_{il}}{t_{hl}^+}\Big)\\
&\qquad - \sum_{h=1}^{m^-}\lambda_h^- b_{hi}s_{ij}\Big(\log\frac{s_{ij}}{t_{hj}^-} - \sum_l s_{il}\log\frac{s_{il}}{t_{hl}^-}\Big)\Bigg)\Big(d\ \log q_{ij}\Big)\\
&= -\tau\sum_{ij}\Bigg(\sum_{h=1}^{m^+}\lambda_h^+ a_{hi}s_{ij}\log\frac{s_{ij}}{t_{hj}^+} - \sum_{h=1}^{m^-}\lambda_h^- b_{hi}s_{ij}\log\frac{s_{ij}}{t_{hj}^-}\\
&\qquad - s_{ij}\sum_l\sum_{h=1}^{m^+}\lambda_h^+ a_{hi}s_{il}\log\frac{s_{il}}{t_{hl}^+}\\
&\qquad + s_{ij}\sum_l\sum_{h=1}^{m^-}\lambda_h^- b_{hi}s_{il}\log\frac{s_{il}}{t_{hl}^-}\Bigg)\Big(d\ \log q_{ij}\Big)\\
&= -\tau\sum_{ij}\left(w_{ij} - s_{ij}\sum_l w_{il}\right)\Big(d\ \log q_{ij}\Big)
\end{aligned}\tag{B.5}$$

where we define

$$\begin{aligned}
w_{ij} &= \left(\sum_{h=1}^{m^+}\lambda_h^+ a_{hi}\right)s_{ij}\log s_{ij} - s_{ij}\sum_{h=1}^{m^+}\lambda_h^+ a_{hi}\log t_{hj}^+\\
&\qquad - \left(\sum_{h=1}^{m^-}\lambda_h^- b_{hi}\right)s_{ij}\log s_{ij} + s_{ij}\sum_{h=1}^{m^-}\lambda_h^- b_{hi}\log t_{hj}^-\\
&= \left(\sum_{h=1}^{m^+}\lambda_h^+ a_{hi} - \sum_{h=1}^{m^-}\lambda_h^- b_{hi}\right)s_{ij}\log s_{ij}\\
&\qquad - s_{ij}\left(\sum_{h=1}^{m^+}\lambda_h^+ a_{hi}\log t_{hj}^+ - \sum_{h=1}^{m^-}\lambda_h^- b_{hi}\log t_{hj}^-\right)
\end{aligned}\tag{B.6}$$

It is interesting to note that

$$
\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{k} w_{ij} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{k}\Bigg(\sum_{h=1}^{m^+} \lambda_h^+ a_{hi} s_{ij}\log s_{ij} - \sum_{h=1}^{m^-} \lambda_h^- b_{hi} s_{ij}\log s_{ij} \\
&\qquad - \sum_{h=1}^{m^+} \lambda_h^+ a_{hi} s_{ij}\log t_{hj}^+ + \sum_{h=1}^{m^-} \lambda_h^- b_{hi} s_{ij}\log t_{hj}^-\Bigg) \\
&= \sum_{h=1}^{m^+} \lambda_h^+ \sum_{i=1}^{n}\sum_{j=1}^{k} a_{hi} s_{ij}\log \frac{s_{ij}}{t_{hj}^+} - \sum_{h=1}^{m^-} \lambda_h^- \sum_{i=1}^{n}\sum_{j=1}^{k} b_{hi} s_{ij}\log \frac{s_{ij}}{t_{hj}^-} \\
&= \sum_{h=1}^{m^+} \lambda_h^+ D_{JS}^+(h) - \sum_{h=1}^{m^-} \lambda_h^- D_{JS}^-(h) = \mathcal{F}(\theta;\mathcal{C})
\end{aligned}
\tag{B.7}
$$

Therefore, summing all w_{ij} provides a way to compute the loss function for constraint violation.

We are now ready to write down the differential of $\mathcal{J}$:

$$
d\ \mathcal{J} = \sum_{i=1}^{n}\sum_{j=1}^{k}\Bigg(\tilde{r}_{ij} - \tau\Big(w_{ij} - s_{ij}\sum_{l=1}^{k} w_{il}\Big)\Bigg)\Big(d\ \log q_{ij}\Big) \tag{B.8}
$$

B.1.2 Gradient Computation

Since the only differentials in Equation (B.8) are $(d\ \log q_{ij})$, the gradient of $\mathcal{J}$ can be obtained by converting these differentials into derivatives. Recall that $q_{ij} = \alpha_j p(\mathbf{y}_i|\theta)$. So,

$$
\frac{\partial}{\partial \log \alpha_j} \log q_{il} = I(j = l),
$$

where $I(.)$ is the indicator function, and is one if the argument is true and zero otherwise. To enforce the restriction that $\alpha_j > 0$ and $\sum_j \alpha_j = 1$, we introduce new variables β_j and express α_j in terms of $\{\beta_j\}$:

$$
\alpha_j = \frac{\exp(\beta_j)}{\sum_{j'=1}^{k} \exp(\beta_{j'})}. \tag{B.9}
$$

We then have

$$\begin{aligned}
\frac{\partial}{\partial \beta_l} \log \alpha_j &= \frac{\partial}{\partial \beta_l} \left(\beta_j - \log \sum_{j'} \exp(\beta_{j'}) \right) = I(j=l) - \frac{\exp(\beta_l)}{\sum_{j'} \exp(\beta_{j'})} \\
&= I(j=l) - \alpha_l \\
\frac{\partial}{\partial \beta_j} \log q_{il} &= \sum_{m=1}^{k} \frac{\partial \log q_{il}}{\partial \log \alpha_m} \frac{\partial \log \alpha_m}{\partial \beta_j} = \sum_m I(l=m) \Big(I(m=j) - \alpha_j \Big) \\
&= I(j=l) - \alpha_j
\end{aligned}$$

If $p(\mathbf{y}_i|\theta_j)$ falls into the exponential family (Section 5.1.1), and θ_j is the natural parameter, the derivative of $\log q_{ij}$ with respect to θ_l can be written as

$$\frac{\partial}{\partial \theta_j} \log q_{il} = I(j=l) \left(\phi(\mathbf{y}_i) - \frac{\partial}{\partial \theta_l} A(\theta_l) \right). \tag{B.10}$$

Note that $\phi(\mathbf{y}_i) - \frac{\partial}{\partial \theta_l} A(\theta_l)$ is zero when the sufficient statistics of the observed data (represented by $\phi(\mathbf{y}_i)$) equal to its expected value (represented by $\frac{\partial}{\partial \theta_l} A(\theta_l)$). In this case, the convexity of $A(\theta_l)$ guarantees that the log-likelihood is maximized.

Before going into the special case of the Gaussian distribution, we want to note that for any number c_{ij}, we have

$$\begin{aligned}
\sum_{ij} c_{ij} \frac{\partial}{\partial \beta_l} \log q_{ij} &= \sum_{ij} c_{ij} \left(I(l=j) - \alpha_l \right) = \sum_i c_{il} - \alpha_l \sum_{ij} c_{ij} \\
\sum_{ij} c_{ij} \frac{\partial}{\partial \theta_l} \log q_{ij} &= \sum_i c_{il} \frac{\partial}{\partial \theta_l} \log q_{il} = \sum_i c_{il} \left(\phi(\mathbf{y}_i) - \frac{\partial}{\partial \theta_l} A(\theta_l) \right) \\
&= \sum_i c_{il} \phi(\mathbf{y}_i) - \frac{\partial}{\partial \theta_l} A(\theta_l) \sum_i c_{il}
\end{aligned}$$

The gradient of $\mathcal{J}$ can be computed by substituting $c_{ij} = \tilde{r}_{ij} - \tau(w_{ij} - s_{ij} \sum_{l=1}^{k} w_{il})$.

B.1.3 Derivative for Gaussian distribution

Consider the special case that $p(\mathbf{y}_i|\theta_l)$ is a Gaussian distribution. Based on Equation (5.6), we can see that the natural parameters are $\boldsymbol{\Upsilon}_l$ and $\boldsymbol{\nu}_l$, the sufficient statistics consist of $\mathbf{y}_i$ and $-\frac{1}{2}\mathbf{y}_i\mathbf{y}_i^T$, and the log-cumulant function $A(\theta_l)$ is given by Equation (5.7). In this case, we have

$$\frac{\partial}{\partial \boldsymbol{\nu}_l} \mathcal{J} = \sum_i c_{il} \mathbf{y}_i - \boldsymbol{\mu}_l \sum_i c_{il} \tag{B.11}$$

$$\frac{\partial}{\partial \boldsymbol{\Upsilon}_l} \mathcal{J} = -\frac{1}{2} \sum_i c_{il} \mathbf{y}_i \mathbf{y}_i^T + \left(\frac{1}{2} \boldsymbol{\mu}_l \boldsymbol{\mu}_l^T + \frac{1}{2} \boldsymbol{\Sigma}_l \right) \sum_i c_{il} \tag{B.12}$$

Note that the above computation implicitly assumes that $\boldsymbol{\Upsilon}_l$ is symmetric. To explicitly enforce the constraints that $\boldsymbol{\Upsilon}_l$ is symmetric and positive definite, we can re-parameterize by its Cholesky decomposition:

$$\boldsymbol{\Upsilon}_l = \mathbf{F}_l \mathbf{F}_l^T, \tag{B.13}$$

Note, however, with this set of parameters, the density is no longer in its natural form. The gradient with respect to $\boldsymbol{\nu}_l$ remains unchanged, and it is not hard to show that

$$\frac{\partial}{\partial \mathbf{F}_l} \log q_{ij} = I(j = l) \left(-\mathbf{y}_i \mathbf{y}_i^T + \boldsymbol{\mu}_l \boldsymbol{\mu}_l^T + \boldsymbol{\Sigma}_l \right) \mathbf{F}_l. \tag{B.14}$$

$$\frac{\partial}{\partial \mathbf{F}_l} \mathcal{J} = -\sum_i c_{il} \mathbf{y}_i \mathbf{y}_i^T \mathbf{F}_l + \left(\boldsymbol{\mu}_l \boldsymbol{\mu}_l^T + \boldsymbol{\Sigma}_l \right) \mathbf{F}_l \sum_i c_{il} \tag{B.15}$$

Alternatively, the Gaussian distribution can be parameterized by the mean $\boldsymbol{\mu}_j$ and the precision matrix $\boldsymbol{\Upsilon}_j$ as in Equation (5.5). Because

$$\frac{\partial}{\partial \boldsymbol{\mu}_l} \log q_{ij} = I(j = l) \boldsymbol{\Upsilon}_l (\mathbf{y}_i - \boldsymbol{\mu}_l) \tag{B.16}$$

$$\frac{\partial}{\partial \boldsymbol{\Upsilon}_l} \log q_{ij} = I(j = l) \left(\frac{1}{2} \boldsymbol{\Sigma}_l - \frac{1}{2} (\mathbf{y}_i - \boldsymbol{\mu}_l)(\mathbf{y}_i - \boldsymbol{\mu}_l)^T \right) \tag{B.17}$$

$$\frac{\partial}{\partial \mathbf{F}_l} \log q_{ij} = I(j = l) \left(\boldsymbol{\Sigma}_l - (\mathbf{y}_i - \boldsymbol{\mu}_l)(\mathbf{y}_i - \boldsymbol{\mu}_l)^T \right) \mathbf{F}_l, \tag{B.18}$$

the corresponding gradient of $\mathcal{J}$ is

$$\frac{\partial}{\partial \boldsymbol{\mu}_l} \mathcal{J} = \boldsymbol{\Upsilon}_l \sum_i c_{il} (\mathbf{y}_i - \boldsymbol{\mu}_l) \tag{B.19}$$

$$\frac{\partial}{\partial \boldsymbol{\Upsilon}_l} \mathcal{J} = \frac{1}{2} \boldsymbol{\Sigma}_l \sum_i c_{il} - \frac{1}{2} \sum_i c_{il} (\mathbf{y}_i - \boldsymbol{\mu}_l)(\mathbf{y}_i - \boldsymbol{\mu}_l)^T \tag{B.20}$$

$$\frac{\partial}{\partial \mathbf{F}_l} \mathcal{J} = \left(\boldsymbol{\Sigma}_l \sum_i c_{il} - \sum_i c_{il} (\mathbf{y}_i - \boldsymbol{\mu}_l)(\mathbf{y}_i - \boldsymbol{\mu}_l)^T \right) \mathbf{F}_l \tag{B.21}$$

B.2 Second Order Information

The second-order information (Hessian) of the proposed objective function $\mathcal{J}$ can be derived in a manner similar to the first order information. We shall first compute the second-order differentials and then convert them to the Hessian matrix. Let $d^2\, x$ denote the second-order differential of the variable x.

B.2.1 Second-order Differential

By taking the differential on both sides of Equation (B.4), we have

$$d^2\, \mathcal{L}^{\text{annealed}}(\theta; \mathcal{Y}, \gamma) = \sum_{ij} (d\, \tilde{r}_{ij}) \left(d\, \log q_{ij} \right) + \sum_{ij} \tilde{r}_{ij} \left(d^2\, \log q_{ij} \right). \tag{B.22}$$

To compute $d\ \tilde{r}_{ij}$, we take the differentials of the logarithm of both sides of Equation (B.3):

$$\begin{aligned} d\ \log \tilde{r}_{ij} &= d\ \left(\log q_{ij}^{\gamma} - \log \sum_{l=1}^{k} q_{il}^{\gamma} \right) \\ &= \gamma\ d\ \log q_{ij} - \frac{1}{\sum_{l'=1}^{k} q_{il'}^{\gamma}} \sum_{l=1}^{k} q_{il}^{\gamma}\ d\ \log q_{il}^{\gamma} \\ &= \gamma \left(d\ \log q_{ij} - \sum_{l=1}^{k} \tilde{r}_{il}\ d\ \log q_{il} \right). \end{aligned} \tag{B.23}$$

Because of the identity that $d\ x = x\ d\ \log x$ for $x > 0$, we have

$$d\ \tilde{r}_{ij} = \tilde{r}_{ij}\ d\ \log \tilde{r}_{ij} \tag{B.24}$$

Substituting Equation (B.24) into Equation (B.22), we have

$$\begin{aligned} & d^2\ \mathcal{L}^{\text{annealed}}(\theta; \mathcal{Y}, \gamma) \\ &= \sum_{ij} \tilde{r}_{ij} \left(d^2\ \log q_{ij} \right) + \gamma \sum_{ij} \tilde{r}_{ij} (d\ \log q_{ij})(d\ \log q_{ij}) \\ &\qquad - \gamma \sum_{i} \sum_{l} \tilde{r}_{il}\ d\ \log q_{il} \sum_{j} \tilde{r}_{ij} d\ \log q_{ij} \\ &= \sum_{ij} \tilde{r}_{ij} \left(d^2\ \log q_{ij} \right) + \gamma \sum_{ijl} (\delta_{jl} - \tilde{r}_{il}) \tilde{r}_{ij} (d\ \log q_{ij})(d\ \log q_{il}). \end{aligned} \tag{B.25}$$

Here, δ_{jl} is the delta function, and it is one if $j = l$ and zero otherwise. The definition of s_{ij} in Equation (5.12) implies the following:

$$d\ s_{ij} = s_{ij}\ d\ \log s_{ij} \tag{B.26}$$

$$d\ \log s_{ij} = \tau \left(d\ \log q_{ij} - \sum_{l=1}^{k} s_{il}\ d\ \log q_{il} \right) \tag{B.27}$$

Note the similarity between the definitions of s_{ij} and $\tilde{r}_{ij}$. Because for any i, $\sum_j (w_{ij} - s_{ij} \sum_l w_{il}) = 0$ and $\sum_j d\ s_{ij} = d\ \sum_j s_{ij} = 0$, Equation (B.5) can be rewritten as

$$\begin{aligned} d\ \mathcal{F}(\theta, \mathcal{C}) &= -\tau \sum_{i} \sum_{j} (w_{ij} - s_{ij} \sum_{l} w_{il})(d\ \log q_{ij}) \\ &= -\tau \sum_{i} \left(\sum_{j} (w_{ij} - s_{ij} \sum_{l} w_{il}) \big(d\ \log q_{ij} - \sum_{l} s_{il}\ d\ \log q_{il} \big) \right) \\ &= -\sum_{i} \left(\sum_{j} (w_{ij} - s_{ij} \sum_{l} w_{il}) d\ \log s_{ij} \right) \\ &= -\sum_{i} \sum_{j} w_{ij}\ d\ \log s_{ij} + \sum_{i} \sum_{j} d\ s_{ij} \sum_{l} w_{il} = -\sum_{i} \sum_{j} w_{ij}\ d\ \log s_{ij} \end{aligned}$$

The differential of this expression yields $d^2\ \mathcal{F}(\theta; \mathcal{C})$. So, we need to find $d\ w_{ij}$ and $d^2\ \log s_{ij}$. The definition of w_{ij} in Equation (B.6) means that its differential is

$$\begin{aligned}
d\ w_{ij} = &\left(\sum_{h=1}^{m^+} \lambda_h^+ a_{hi} - \sum_{h=1}^{m^-} \lambda_h^- b_{hi}\right)(\log s_{ij} + 1) s_{ij}\ d\ \log s_{ij} \\
&\qquad - s_{ij} \left(\sum_{h=1}^{m^+} \frac{\lambda_h^+ a_{hi}}{t_{hj}^+} d\ t_{hj}^+ - \sum_{h=1}^{m^-} \frac{\lambda_h^- b_{hi}}{t_{hj}^-} d\ t_{hj}^-\right) \\
&\qquad\qquad - s_{ij} \left(\sum_{h=1}^{m^+} \lambda_h^+ a_{hi} \log t_{hj}^+ - \sum_{h=1}^{m^-} \lambda_h^- b_{hi} \log t_{hj}^-\right) d\ \log s_{ij} \\
= &\ w_{ij}^* d\ \log s_{ij} - \sum_{h=1}^{m^+} \frac{\lambda_h^+}{t_{hj}^+} a_{hi} s_{ij} (d\ t_{hj}^+) + \sum_{h=1}^{m^-} \frac{\lambda_h^-}{t_{hj}^-} b_{hi} s_{ij} (d\ t_{hj}^-),
\end{aligned}$$

where we define $w_{ij}^* = w_{ij} + \left(\sum_{h=1}^{m^+} \lambda_h^+ a_{hi} - \sum_{h=1}^{m^-} \lambda_h^- b_{hi}\right) s_{ij}$. Taking the differentials of both sides of Equation (B.27), we have

$$\begin{aligned}
d^2\ \log s_{ij} &= \tau \Big(d^2\ \log q_{ij} - \sum_{l=1}^{k} s_{il}\ d^2\ \log q_{il} - \sum_{l=1}^{k} s_{il} (d\ \log s_{il})(d\ \log q_{il}) \Big) \\
&= \tau \Big(d^2\ \log q_{ij} - \sum_{l=1}^{k} s_{il}\ d^2\ \log q_{il} \Big) \\
&\qquad - \sum_{l=1}^{k} s_{il} (d\ \log s_{il}) \big(d\ \log s_{il} + \tau \sum_{l'=1}^{k} s_{il'}\ d\ \log q_{il'} \big) \\
&= \tau \Big(d^2\ \log q_{ij} - \sum_{l=1}^{k} s_{il}\ d^2\ \log q_{il} \Big) - \sum_{l=1}^{k} s_{il} (d\ \log s_{il})(d\ \log s_{il})
\end{aligned}$$

Note that we have used the fact $\sum_{l=1}^{k} s_{il} (d\ \log s_{il}) = 0$. If we define $\tilde{w}_{ij} = w_{ij} - s_{ij} \sum_{l=1}^{k} w_{il}$, we can write

$$\begin{aligned}
&\sum_{ij} w_{ij}\ d^2\ \log s_{ij} \\
&= \tau \sum_{ij} \tilde{w}_{ij} d^2\ \log q_{ij} - \sum_{i} \sum_{j=1}^{k} w_{ij} \sum_{l=1}^{k} s_{il} (d\ \log s_{il})(d\ \log s_{il}),
\end{aligned} \tag{B.28}$$

Putting them together, we have

$$\begin{aligned}
d^2\ \mathcal{F}(\theta;\mathcal{C}) &= \sum_{ij} w_{ij}\ d^2\ \log s_{ij} + \sum_{ij} d\ w_{ij}\ d\ \log s_{ij} \\
&= \sum_{ij} w_{ij}\ d^2\ \log s_{ij} + \sum_{ij} w^*_{ij}(d\ \log s_{ij})(d\ \log s_{ij}) \\
&\quad - \sum_j \sum_{h=1}^{m^+} \sum_i \frac{\lambda^+_h}{t^+_{hj}} (d\ t^+_{hj}) \big(a_{hi} s_{ij}\ d\ \log s_{ij}\big) \\
&\quad + \sum_j \sum_{h=1}^{m^-} \sum_i \frac{\lambda^-_h}{t^-_{hj}} (d\ t^-_{hj}) \big(b_{hi} s_{ij}\ d\ \log s_{ij}\big) \\
&= \tau \sum_{ij} \tilde{w}_{ij}\ d^2\ \log q_{ij} + \sum_{ij} \big(w^*_{ij} - s_{ij} \sum_{l=1}^{k} w_{il}\big)(d\ \log s_{ij})(d\ \log s_{ij}) \\
&\quad - \sum_{h=1}^{m^+} \sum_j \frac{\lambda^+_h}{t^+_{hj}} (d\ t^+_{hj})(d\ t^+_{hj}) + \sum_{h=1}^{m^-} \sum_j \frac{\lambda^-_h}{t^-_{hj}} (d\ t^-_{hj})(d\ t^-_{hj})
\end{aligned} \tag{B.29}$$

Note that $w^*_{ij} - s_{ij}\sum_{l=1}^{k} w_{il} = \tilde{w}_{ij} + \left(\sum_{h=1}^{m^+} \lambda^+_h a_{hi} - \sum_{h=1}^{m^-} \lambda^-_h b_{hi}\right) s_{ij}$.

B.2.2 Obtaining the Hessian matrix

Our goal in this section is to obtain the Hessian matrix of $\mathcal{J}$ with respect to the parameters ordered by $\beta_1, \ldots, \beta_k, \ldots \theta_1, \ldots, \theta_k$. Let $\boldsymbol{\psi}_{iu}$ denote the column vector $\dfrac{\partial \log q_{ij}}{\partial \theta_u}$, and define $\boldsymbol{\Psi}_u$ to be the $|\theta_u|$ by n matrix $[\boldsymbol{\psi}_{1u}, \ldots, \boldsymbol{\psi}_{nu}]$, where $|\theta_u|$ is the number of parameters in θ_u. Let $\mathbf{D}_{uv}$ be a n by n diagonal matrix such that its (i,i)-th entry is $\gamma(\delta_{uv} - \tilde{r}_{iv})\tilde{r}_{iu}$. Let $\mathbf{1}_{1,n}$ denote a 1 by n matrix with all its entries equal to one. Let $\mathbf{H}_\beta$ be the Hessian matrix of $(\log q_{ij})$ with respect to the β_us, i.e., the (u,v)-th entry of $\mathbf{H}_\beta$ is given by

$$\frac{\partial^2}{\partial \beta_u \partial \beta_v} \log q_{ij} = \frac{\partial^2}{\partial \beta_u \partial \beta_v} \log \alpha_j = \frac{\partial}{\partial \beta_v}\Big(\delta_{ju} - \alpha_u\Big) = -\alpha_u\left(\delta_{uv} - \alpha_v\right).$$

Here, δ_{uv} is the Kronecker delta, which is 1 if $u = v$ and 0 otherwise. Note that $\mathbf{H}_\beta$ does not depend on the value of i and j in $\log q_{ij}$. Let $\mathbf{H}_{ij}$ denote the Hessian of $\log p(\mathbf{y}_i|\theta_j)$ with respect to θ_j. Its exact form for the case of Gaussian distributions will be derived in the next section.

B.2.2.1 Hessian of the Log-likelihood

Based on Equation (B.25), we have

$$
\begin{aligned}
&\frac{\partial^2}{\partial\theta_u\partial\theta_v}\mathcal{L}^{\text{annealed}}(\theta;\mathcal{Y},\gamma)\\
&=\sum_{ij}\tilde{r}_{ij}\frac{\partial^2\log q_{ij}}{\partial\theta_u\partial\theta_v}+\gamma\sum_{ijl}(\delta_{jl}-\tilde{r}_{il})\tilde{r}_{ij}\left(\frac{\partial\log q_{ij}}{\partial\theta_u}\right)\left(\frac{\partial\log q_{il}}{\partial\theta_v}\right)^T\\
&=\delta_{uv}\sum_i\tilde{r}_{iu}\frac{\partial^2\log q_{iu}}{\partial\theta_u^2}+\gamma\sum_i(\delta_{uv}-\tilde{r}_{iv})\tilde{r}_{iu}\boldsymbol{\psi}_{iu}\boldsymbol{\psi}_{iv}^T\\
&=\delta_{uv}\sum_i\tilde{r}_{iu}\mathbf{H}_{iu}+\boldsymbol{\Psi}_u\mathbf{D}_{uv}\boldsymbol{\Psi}_v^T\\
&\frac{\partial^2}{\partial\beta_u\partial\theta_v}\mathcal{L}^{\text{annealed}}(\theta;\mathcal{Y},\gamma)\\
&=\gamma\sum_{ijl}(\delta_{jl}-\tilde{r}_{il})\tilde{r}_{ij}\Big(\delta_{uj}-\alpha_u\Big)\delta_{lv}\left(\frac{\partial\log q_{il}}{\partial\theta_v}\right)^T=\gamma\sum_i(\delta_{uv}-\tilde{r}_{iv})\tilde{r}_{iu}\boldsymbol{\psi}_{iv}^T\\
&=\mathbf{1}_{1,n}\mathbf{D}_{uv}\boldsymbol{\Psi}_v^T\\
&\frac{\partial^2}{\partial\beta_u\partial\beta_v}\mathcal{L}^{\text{annealed}}(\theta;\mathcal{Y},\gamma)\\
&=\sum_{ij}\tilde{r}_{ij}\alpha_u(\alpha_v-\delta_{uv})+\gamma\sum_{ijl}(\delta_{jl}-\tilde{r}_{il})\tilde{r}_{ij}(\delta_{uj}-\alpha_u)(\delta_{vl}-\alpha_v)\\
&=n\alpha_u(\alpha_v-\delta_{uv})+\gamma\sum_i(\delta_{uv}-\tilde{r}_{iv})\tilde{r}_{iu}=n\alpha_u(\alpha_v-\delta_{uv})+\mathbf{1}_{1,n}\mathbf{D}_{uv}\mathbf{1}_{1,n}^T
\end{aligned}
$$

Define $\tilde{\mathbf{H}}^{\mathcal{L}}$, the "expected Hessian" of the annealed log-likelihood, by

$$\tilde{\mathbf{H}}^{\mathcal{L}}=\text{blk-diag}(n\mathbf{H}_\beta,\sum_i\tilde{r}_{i1}\mathbf{H}_{i1},\ldots,\sum_i\tilde{r}_{ik}\mathbf{H}_{ik}). \tag{B.30}$$

It can be viewed as the expected value of the Hessian matrix of the complete-data log-likelihood. Define a $k(1+|\theta_1|)$ by nk matrix $\boldsymbol{\Lambda}$ and partition it into $2k$ by k blocks, so that the (j,j)-th block is $\mathbf{1}_{1,n}$, and the $(j+k,j)$-th block is $\boldsymbol{\Psi}_j$, where $1\le i\le k$. All other entries in $\boldsymbol{\Lambda}$ are zero. In other words,

$$\boldsymbol{\Lambda}=\begin{bmatrix}\mathbf{1}_{1,n}&&\\&\ddots&\\&&\mathbf{1}_{1,n}\\\boldsymbol{\Psi}_1&&\\&\ddots&\\&&\boldsymbol{\Psi}_k\end{bmatrix}=\begin{bmatrix}\mathbf{1}_{1,n}&&\\&\ddots&\\&&\mathbf{1}_{1,n}\\{[\boldsymbol{\psi}_{11}\cdots\boldsymbol{\psi}_{n1}]}&&\\&\ddots&\\&&{[\boldsymbol{\psi}_{1k}\cdots\boldsymbol{\psi}_{nk}]}\end{bmatrix} \tag{B.31}$$

Let $\mathbf{D}$ be a nk by nk matrix and we partition it into k by k blocks, so that the (u,v)-th block is $\mathbf{D}_{uv}$. With these notation, the Hessian of the annealed log-likelihood is

$$\frac{\partial^2}{\partial\theta^2}\mathcal{L}^{\text{annealed}}(\theta;\mathcal{Y},\gamma)=\tilde{\mathbf{H}}^{\mathcal{L}}+\boldsymbol{\Lambda}\mathbf{D}\boldsymbol{\Lambda}^T \tag{B.32}$$

$\mathbf{D}$ is symmetric because the (i, i)-th element of both $\mathbf{D}_{uv}$ and $\mathbf{D}_{vu}$ are $\gamma(\delta_{uv} - \tilde{r}_{iu})\tilde{r}_{iv}$. Also, the sum of each of the column of $\mathbf{D}$ is 0 because $\sum_u \gamma(\delta_{uv} - \tilde{r}_{iu})\tilde{r}_{iv} = 0$.

B.2.2.2 Hessian of the Constraint Violation Term

By converting the differentials in Equation (B.27) into derivatives, we have

$$\frac{\partial \log s_{ij}}{\partial \theta_u} = \tau\Big(\delta_{ju}\boldsymbol{\psi}_{iu} - \sum_{l=1}^{k} s_{il}\delta_{lu}\boldsymbol{\psi}_{iu}\Big) = \tau(\delta_{ju} - s_{iu})\boldsymbol{\psi}_{iu}$$

$$\frac{\partial \log s_{ij}}{\partial \beta_u} = \tau\Big(\delta_{ju} - \alpha_u - \sum_{l=1}^{k} s_{il}(\delta_{lu} - \alpha_u)\Big) = \tau(\delta_{ju} - s_{iu})$$

This implies

$$\sum_{ij}(w^*_{ij} - s_{ij}\sum_{l=1}^{k} w_{il})\Big(\frac{\partial \log s_{ij}}{\partial \beta_u}\Big)\Big(\frac{\partial \log s_{ij}}{\partial \beta_v}\Big)^T$$
$$= \tau^2 \sum_i \left(\sum_j (w^*_{ij} - s_{ij}\sum_{l=1}^{k} w_{il})(\delta_{ju} - s_{iu})(\delta_{jv} - s_{iv})\right) = \mathbf{1}_{1,n}\mathbf{E}_{uv}\mathbf{1}_{1,n}^T$$

Similarly, we have

$$\sum_{ij}(w^*_{ij} - s_{ij}\sum_{l=1}^{k} w_{il})\Big(\frac{\partial \log s_{ij}}{\partial \beta_u}\Big)\Big(\frac{\partial \log s_{ij}}{\partial \theta_v}\Big)^T = \mathbf{1}_{1,n}\mathbf{E}_{uv}\boldsymbol{\Psi}_v^T$$

$$\sum_{ij}(w^*_{ij} - s_{ij}\sum_{l=1}^{k} w_{il})\Big(\frac{\partial \log s_{ij}}{\partial \theta_u}\Big)\Big(\frac{\partial \log s_{ij}}{\partial \theta_v}\Big)^T = \boldsymbol{\Psi}_u\mathbf{E}_{uv}\boldsymbol{\Psi}_v^T$$

Here, $\tau^2\left(\sum_j (w^*_{ij} - s_{ij}\sum_{l=1}^{k} w_{il})(\delta_{ju} - s_{iu})(\delta_{jv} - s_{iv})\right)$ is the (i, i)-th element of the n by n diagonal matrix $\mathbf{E}_{uv}$. Let $\mathbf{a}_{hju}$ denote a vector of length n such that its i-th entry is given by $\tau a_{hi}s_{ij}(\delta_{ju} - s_{iu})$. Because $d\ t^+_{hj} = \sum_i a_{hi}\ d\ s_{ij} = \sum_i a_{hi}s_{ij}\ d\ \log s_{ij}$, we have

$$\frac{\partial t^+_{hj}}{\partial \theta_u} = \sum_i a_{hi}s_{ij}\frac{\partial \log s_{ij}}{\partial \theta_u} = \tau\sum_i a_{hi}s_{ij}(\delta_{ju} - s_{iu})\boldsymbol{\psi}_{iu} = \boldsymbol{\Psi}_u\mathbf{a}_{hju}$$

$$\frac{\partial t^+_{hj}}{\partial \beta_u} = \tau\sum_i a_{hi}s_{ij}(\delta_{ju} - s_{iu}) = \mathbf{1}_{1,n}\mathbf{a}_{hju}$$

This means that

$$\sum_{h=1}^{m^+}\sum_{j=1}^{k}\frac{\lambda^+_h}{t^+_{hj}}\left(\frac{\partial t^+_{hj}}{\partial \theta_u}\right)\left(\frac{\partial t^+_{hj}}{\partial \theta_v}\right)^T = \boldsymbol{\Psi}_u\left(\sum_{h=1}^{m^+}\sum_{j=1}^{k}\frac{\lambda^+_h}{t^+_{hj}}\mathbf{a}_{hju}\mathbf{a}^T_{hjv}\right)\boldsymbol{\Psi}^T_v$$
$$= \boldsymbol{\Psi}_u\mathbf{A}_u\mathbf{L}^+\mathbf{A}^T_v\boldsymbol{\Psi}^T_v,$$

where we concatenate different $\mathbf{a}_{hju}$ to form a n by km^+ matrix $\mathbf{A}_u$, defined by

$$\mathbf{A}_u = [\mathbf{a}_{1,1,u}, \mathbf{a}_{2,1,u}, \ldots, \mathbf{a}_{m^+,1,u}, \mathbf{a}_{1,2,u}, \ldots, \mathbf{a}_{m^+,2,u}, \ldots, \mathbf{a}_{1,k,u}, \ldots, \mathbf{a}_{m^+,k,u}].$$

Note that $\mathbf{A}_u$ has similar sparsity pattern as the matrix $\{a_{hi}\}$. The diagonal matrix $\mathbf{L}^+$ is of size km^+ by km^+. Its diagonal entries are given by $\frac{\lambda_h^+}{t_{hj}^+}$, and the ordering of these diagonal entries matches the ordering of $\mathbf{a}_{hju}$ in $\mathbf{A}_u$. By similar reasoning, we have

$$\sum_{h=1}^{m^+}\sum_{j=1}^{k} \frac{\lambda_h^+}{t_{hj}^+} \left(\frac{\partial t_{hj}^+}{\partial \beta_u}\right) \left(\frac{\partial t_{hj}^+}{\partial \theta_v}\right)^T = \mathbf{1}_{1,n}\mathbf{A}_u\mathbf{L}^+\mathbf{A}_v^T\mathbf{\Psi}_v^T$$

$$\sum_{h=1}^{m^+}\sum_{j=1}^{k} \frac{\lambda_h^+}{t_{hj}^+} \left(\frac{\partial t_{hj}^+}{\partial \beta_u}\right) \left(\frac{\partial t_{hj}^+}{\partial \beta_v}\right) = \mathbf{1}_{1,n}\mathbf{A}_u\mathbf{L}^+\mathbf{A}_v^T\mathbf{1}_{1,n}^T$$

The case for t_{hj}^-, which corresponds to to must-not-link constraints, is similar. So, we define $\mathbf{b}_{hju}$ to consist of $\tau b_{hi} s_{ij}(\delta_{ju} - s_{iu})$ for different i, and concatenate $\mathbf{b}_{hju}$ to form $\mathbf{B}_u$. $\mathbf{L}^-$ is a diagonal matrix with entries $\frac{\lambda_h^-}{t_{hj}^-}$. Substituting all these into the result derived in Equation (B.29), we have

$$\begin{aligned}
&\frac{\partial}{\partial \theta_u}\sum_{ij} w_{ij}\frac{\partial}{\partial \theta_v}\log s_{ij} \\
&= \tau\sum_{ij}\tilde{w}_{ij}\frac{\partial^2}{\partial\theta_u\partial\theta_v}\log q_{ij} + \sum_{ij}(w_{ij}^* - s_{ij}\sum_{l=1}^{k} w_{il})(\frac{\partial}{\partial\theta_u}\log s_{ij})(\frac{\partial}{\partial\theta_v}\log s_{ij})^T \\
&\quad - \sum_{h=1}^{m^+}\sum_j \frac{\lambda_h^+}{t_{hj}^+}(\frac{\partial}{\partial\theta_u}t_{hj}^+)(\frac{\partial}{\partial\theta_v}t_{hj}^+)^T + \sum_{h=1}^{m^-}\sum_j \frac{\lambda_h^-}{t_{hj}^-}(\frac{\partial}{\partial\theta_u}t_{hj}^-)(\frac{\partial}{\partial\theta_v}t_{hj}^-)^T \\
&= \tau\delta_{uv}\sum_i \tilde{w}_{iu}\frac{\partial^2 \log q_{iu}}{\partial\theta_u^2} + \mathbf{\Psi}_u\mathbf{E}_{uv}\mathbf{\Psi}_v^T - \mathbf{\Psi}_u\mathbf{A}_u\mathbf{L}^+\mathbf{A}_v^T\mathbf{\Psi}_v^T + \mathbf{\Psi}_u\mathbf{B}_u\mathbf{L}^-\mathbf{B}_v^T\mathbf{\Psi}_v^T \\
&= \tau\delta_{uv}\sum_i \tilde{w}_{iu}\frac{\partial^2 \log q_{iu}}{\partial\theta_u^2} + \mathbf{\Psi}_u\left(\mathbf{E}_{uv} - \mathbf{A}_u\mathbf{L}^+\mathbf{A}_v^T + \mathbf{B}_u\mathbf{L}^-\mathbf{B}_v^T\right)\mathbf{\Psi}_v^T
\end{aligned}$$

Similarly, we have

$$\frac{\partial}{\partial \beta_u}\sum_{ij} w_{ij}\frac{\partial}{\partial \theta_v}\log s_{ij} = \mathbf{1}_{1,n}\left(\mathbf{E}_{uv} - \mathbf{A}_u\mathbf{L}^+\mathbf{A}_v^T + \mathbf{B}_u\mathbf{L}^-\mathbf{B}_v^T\right)\mathbf{\Psi}_v^T$$

$$\begin{aligned}
\frac{\partial}{\partial \beta_u}\sum_{ij} w_{ij}\frac{\partial}{\partial \beta_v}\log s_{ij} &= \tau\sum_{ij}\tilde{w}_{ij}\frac{\partial^2 \log q_{ij}}{\partial\beta_u\partial\beta_v} + \\
&\qquad \mathbf{1}_{1,n}\left(\mathbf{E}_{uv} - \mathbf{A}_u\mathbf{L}^+\mathbf{A}_v^T + \mathbf{B}_u\mathbf{L}^-\mathbf{B}_v^T\right)\mathbf{1}_{1,n}^T
\end{aligned}$$

Let $\mathbf{H}^{\mathcal{C}}$ denote the "expected" hessian of the complete data log-likelihood due to the constraints, i.e.,

$$\tilde{\mathbf{H}}^{\mathcal{C}} = \text{blk-diag}(\mathbf{0}, \tau \sum_i \tilde{w}_{i1}\mathbf{H}_{i1}, \ldots, \tau \sum_i \tilde{w}_{ik}\mathbf{H}_{ik}). \tag{B.33}$$

Note that there are no Hessian terms corresponding to the β_j because $\sum_j \tilde{w}_{ij} = 0$. Let $\mathbf{E}$ be a nk by nk matrix. We partition it into k by k blocks, such that the (u,v)-th block is $\mathbf{E}_{uv}$. Let $\mathbf{A}$ be a nk by km^+ matrix and $\mathbf{B}$ be a nk by km^- matrix, such that

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_k \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \vdots \\ \mathbf{B}_k \end{bmatrix}$$

We are now ready to state the Hessian term corresponding to the constraints:

$$\begin{aligned} &\frac{\partial^2}{\partial \theta^2}\left(-\sum_{h=1}^{m^+} \lambda_h^+ D_{JS}^+(h) + \sum_{h=1}^{m^-} \lambda_h^- D_{JS}^-(h) \right) \\ &\qquad = -\tilde{\mathbf{H}}^{\mathcal{C}} - \mathbf{\Lambda}\mathbf{E}\mathbf{\Lambda}^T + \mathbf{\Lambda}\mathbf{A}\mathbf{L}^+\mathbf{A}^T\mathbf{\Lambda}^T - \mathbf{\Lambda}\mathbf{B}\mathbf{L}^-\mathbf{B}^T\mathbf{\Lambda}^T \end{aligned} \tag{B.34}$$

Note that the sum of each of the columns of $\mathbf{A}$ is 0, because $\sum_{iu} \tau a_{hi} s_{ij}(\delta_{ju} - s_{iu}) = \sum_i \tau a_{hi} s_{ij} \sum_u (\delta_{ju} - s_{iu}) = 0$. Combine Equation (B.34) with Equation (B.32), we have the Hessian of the objective function $\mathcal{J}$ in matrix form:

$$\begin{aligned} \frac{\partial^2}{\partial \theta^2}\mathcal{J} &= \tilde{\mathbf{H}}^{\mathcal{L}} - \tilde{\mathbf{H}}^{\mathcal{C}} + \mathbf{\Lambda}\mathbf{D}\mathbf{\Lambda}^T - \mathbf{\Lambda}\mathbf{E}\mathbf{\Lambda}^T + \mathbf{\Lambda}\mathbf{A}\mathbf{L}^+\mathbf{A}^T\mathbf{\Lambda}^T - \mathbf{\Lambda}\mathbf{B}\mathbf{L}^-\mathbf{B}^T\mathbf{\Lambda}^T \\ &= \tilde{\mathbf{H}}^{\mathcal{LC}} + \mathbf{\Lambda}\left(\mathbf{D} - \mathbf{E} + \mathbf{A}\mathbf{L}^+\mathbf{A}^T - \mathbf{B}\mathbf{L}^-\mathbf{B}^T\right)\mathbf{\Lambda}^T. \end{aligned} \tag{B.35}$$

Here, $\tilde{\mathbf{H}}^{\mathcal{LC}} = \tilde{\mathbf{H}}^{\mathcal{L}} - \tilde{\mathbf{H}}^{\mathcal{C}}$ is the combined expected Hessian.

B.2.3 Hessian of the Gaussian Probability Density Function

Computation of $\tilde{\mathbf{H}}^{\mathcal{LC}}$ requires $\mathbf{H}_{ij}$, which is the result of differentiating $\log p(\mathbf{y}_i|\theta_j)$ with respect to the parameter θ_j twice. We shall derive the explicit form of $\mathbf{H}_{ij}$ when $\log p(\mathbf{y}_i|\theta_j)$ is the Gaussian pdf. For simplicity, we shall omit the reference to the object index i and the cluster index j in our derivation.

We shall need some notations in matrix calculus [179] in our derivation. Let $\operatorname{vec}\mathbf{X}$ denote a vector of length pq formed by stacking the columns of a p by q matrix $\mathbf{X}$. Let $\mathbf{Y}$ be a r by s matrix. The Kronecker product $\mathbf{X} \otimes \mathbf{Y}$ is a pr by qs matrix defined by

$$\mathbf{X} \otimes \mathbf{Y} = \begin{bmatrix} x_{11}\mathbf{Y} & x_{12}\mathbf{Y} & \ldots & x_{1q}\mathbf{Y} \\ x_{21}\mathbf{Y} & x_{22}\mathbf{Y} & \ldots & x_{2q}\mathbf{Y} \\ \vdots & & & \vdots \\ x_{p1}\mathbf{Y} & & \ldots & x_{pq}\mathbf{Y} \end{bmatrix}. \tag{B.36}$$

The precedence of the operator $\otimes$ is defined to be lower than matrix multiplication, i.e., $\mathbf{XY} \otimes \mathbf{Z}$ is the same as $(\mathbf{XY}) \otimes \mathbf{Z}$. The following identity is used frequently in this section:

$$\operatorname{vec}(\mathbf{XYZ}) = (\mathbf{Z}^T \otimes \mathbf{X}) \operatorname{vec} \mathbf{Y}. \tag{B.37}$$

Let $\mathbf{K}_d$ denote a permutation matrix of size d^2 by d^2, such that

$$\mathbf{K}_d \operatorname{vec} \mathbf{Z} = \operatorname{vec} \mathbf{Z}^T, \tag{B.38}$$

where $\mathbf{Z}$ is a d by d matrix. Note that $\mathbf{K}_d^T = \mathbf{K}_d^{-1} = \mathbf{K}_d$.

B.2.3.1 Natural Parameter

When the density is parameterized by its natural parameter as in Equation (5.6), we have

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\nu}} \log p(\mathbf{y}) &= \mathbf{y} - \frac{1}{2}\left(\boldsymbol{\Upsilon}^{-1} + \boldsymbol{\Upsilon}^{-T}\right)\boldsymbol{\nu} \\
\frac{\partial}{\partial \boldsymbol{\Upsilon}} \log p(\mathbf{y}) &= -\frac{1}{2}\mathbf{y}\mathbf{y}^T + \frac{1}{2}\boldsymbol{\Upsilon}^{-T} + \frac{1}{2}\boldsymbol{\Upsilon}^{-T}\boldsymbol{\nu}\boldsymbol{\nu}^T\boldsymbol{\Upsilon}^{-T} \\
\frac{\partial}{\partial \mathbf{F}} \log p(\mathbf{y}) &= -\mathbf{y}\mathbf{y}^T\mathbf{F} + \mathbf{F}^{-T} + \boldsymbol{\Upsilon}^{-T}\boldsymbol{\nu}\boldsymbol{\nu}^T\boldsymbol{\Upsilon}^{-T}\mathbf{F},
\end{aligned}$$

where $\boldsymbol{\Upsilon}^{-T}$ denotes the transpose of $\boldsymbol{\Upsilon}^{-1}$. Therefore,

$$\begin{aligned}
\frac{\partial^2}{\partial \boldsymbol{\nu}^2} \log p(\mathbf{y}) &= -\frac{1}{2}\left(\boldsymbol{\Upsilon}^{-1} + \boldsymbol{\Upsilon}^{-T}\right) \\
\frac{\partial^2}{\partial \operatorname{vec}\boldsymbol{\Upsilon}\ \partial \boldsymbol{\nu}} \log p(\mathbf{y}) &= \frac{1}{2}\left(\boldsymbol{\Upsilon}^{-1} \otimes \boldsymbol{\Upsilon}^{-T}\boldsymbol{\nu} + \boldsymbol{\Upsilon}^{-1}\boldsymbol{\nu} \otimes \boldsymbol{\Upsilon}^{-T}\right) \\
&= \frac{1}{2}\left(\boldsymbol{\Upsilon}^{-1} \otimes \boldsymbol{\Upsilon}^{-T}\right)(\mathbf{I}_d \otimes \boldsymbol{\nu} + \boldsymbol{\nu} \otimes \mathbf{I}_d) \\
\frac{\partial^2}{\partial \operatorname{vec}\mathbf{F}\ \partial \boldsymbol{\nu}} \log p(\mathbf{y}) &= \mathbf{F}^T\boldsymbol{\Upsilon}^{-1} \otimes \boldsymbol{\Upsilon}^{-T}\boldsymbol{\nu} + \mathbf{F}^T\boldsymbol{\Upsilon}^{-1}\boldsymbol{\nu} \otimes \boldsymbol{\Upsilon}^{-T} \\
&= \left(\mathbf{F}^{-1} \otimes \boldsymbol{\Sigma}\right)(\mathbf{I}_d \otimes \boldsymbol{\nu} + \boldsymbol{\nu} \otimes \mathbf{I}_d)
\end{aligned}$$

The last term in the Hessian matrix requires more work. We first take the differential with respect to $\boldsymbol{\Upsilon}$:

$$\begin{aligned}
d\,\frac{\partial}{\partial \boldsymbol{\Upsilon}} \log p(\mathbf{y}) = &-\frac{1}{2}\boldsymbol{\Upsilon}^{-T}(d\ \boldsymbol{\Upsilon}^T)\boldsymbol{\Upsilon}^{-T} \\
&-\frac{1}{2}\boldsymbol{\Upsilon}^{-T}\boldsymbol{\nu}\boldsymbol{\nu}^T\boldsymbol{\Upsilon}^{-T}(d\ \boldsymbol{\Upsilon}^T)\boldsymbol{\Upsilon}^{-T} - \frac{1}{2}\boldsymbol{\Upsilon}^{-T}(d\ \boldsymbol{\Upsilon}^T)\boldsymbol{\Upsilon}^{-T}\boldsymbol{\nu}\boldsymbol{\nu}^T\boldsymbol{\Upsilon}^{-T}.
\end{aligned}$$

By using the identity in Equation (B.37), the Hessian term can be obtained as

$$\begin{aligned}
\frac{\partial^2}{\partial (\operatorname{vec}\boldsymbol{\Upsilon})^2} \log p(\mathbf{y}) = &-\frac{1}{2}\left(\boldsymbol{\Upsilon}^{-1} \otimes \boldsymbol{\Upsilon}^{-T}\right)\mathbf{K}_d \\
&-\frac{1}{2}\left(\boldsymbol{\Upsilon}^{-1} \otimes \boldsymbol{\Upsilon}^{-T}\boldsymbol{\nu}\boldsymbol{\nu}^T\boldsymbol{\Upsilon}^{-T}\right)\mathbf{K}_d - \frac{1}{2}\left(\boldsymbol{\Upsilon}^{-1}\boldsymbol{\nu}\boldsymbol{\nu}^T\boldsymbol{\Upsilon}^{-1} \otimes \boldsymbol{\Upsilon}^{-T}\right)\mathbf{K}_d
\end{aligned}$$

Similarly, the Hessian term corresponding to $\mathbf{F}$ can be obtained if we note that

$$\begin{aligned}
d\,\frac{\partial}{\partial \mathbf{F}} \log p(\mathbf{y}) &= -\mathbf{y}\mathbf{y}^T(d\,\mathbf{F}) - \mathbf{F}^{-T}(d\,\mathbf{F}^T)\mathbf{F}^{-T} - \boldsymbol{\Upsilon}^{-T}\boldsymbol{\nu}\boldsymbol{\nu}^T\mathbf{F}^{-T}(d\,\mathbf{F}^T)\mathbf{F}^{-T} \\
&\quad - \boldsymbol{\Upsilon}^{-T}(\mathbf{F}(d\,\mathbf{F}^T) + (d\,\mathbf{F})\mathbf{F}^T)\boldsymbol{\Upsilon}^{-T}\boldsymbol{\nu}\boldsymbol{\nu}^T\mathbf{F}^{-T} \\
\frac{\partial^2}{\partial(\operatorname{vec}\mathbf{F})^2} \log p(\mathbf{y}) &= -\mathbf{I}_d \otimes \mathbf{y}\mathbf{y}^T - \left(\mathbf{F}^{-1} \otimes \mathbf{F}^{-T}\right)\mathbf{K}_d - \left(\mathbf{F}^{-1} \otimes \boldsymbol{\mu}\boldsymbol{\mu}^T\mathbf{F}\right)\mathbf{K}_d \\
&\quad - \left(\mathbf{F}^T\boldsymbol{\mu}\boldsymbol{\mu}^T \otimes \mathbf{F}^{-T}\right)\mathbf{K}_d - \mathbf{F}^T\boldsymbol{\mu}\boldsymbol{\mu}^T\mathbf{F} \otimes \boldsymbol{\Sigma}
\end{aligned}$$

In the special case that $\boldsymbol{\Upsilon}$ is always symmetric, we can have a simpler Hessian term. This amounts to assuming that $\boldsymbol{\Upsilon}^T = \boldsymbol{\Upsilon}$ and $(d\boldsymbol{\Upsilon})^T = (d\boldsymbol{\Upsilon})$. We have

$$\begin{aligned}
\frac{\partial^2}{\partial(\operatorname{vec}\boldsymbol{\Upsilon})^2} \log p(\mathbf{y}) &= -\frac{1}{2}\boldsymbol{\Sigma} \otimes \boldsymbol{\Sigma} - \frac{1}{2}\boldsymbol{\Sigma} \otimes \boldsymbol{\mu}\boldsymbol{\mu}^T - \frac{1}{2}\boldsymbol{\mu}\boldsymbol{\mu}^T \otimes \boldsymbol{\Sigma} \\
&= -\frac{1}{2}\left((\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T) \otimes (\boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^T) - (\boldsymbol{\mu} \otimes \boldsymbol{\mu})(\boldsymbol{\mu}^T \otimes \boldsymbol{\mu}^T)\right)
\end{aligned}$$

B.2.3.2 Moment Parameter

When moment parameter is used as in Equation (5.6) for the density, we have

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\mu}} \log p(\mathbf{y}) &= \frac{1}{2}\left(\boldsymbol{\Upsilon} + \boldsymbol{\Upsilon}^T\right)(\mathbf{y} - \boldsymbol{\mu}) \\
\frac{\partial}{\partial \boldsymbol{\Upsilon}} \log p(\mathbf{y}) &= -\frac{1}{2}(\mathbf{y} - \boldsymbol{\mu})(\mathbf{y} - \boldsymbol{\mu})^T + \frac{1}{2}\boldsymbol{\Upsilon}^{-T} \\
\frac{\partial}{\partial \mathbf{F}} \log p(\mathbf{y}) &= -(\mathbf{y} - \boldsymbol{\mu})(\mathbf{y} - \boldsymbol{\mu})^T\mathbf{F} + \mathbf{F}^{-T}
\end{aligned}$$

The second-order terms include

$$\begin{aligned}
\frac{\partial^2}{\partial \boldsymbol{\mu}^2} \log p(\mathbf{y}) &= -\frac{1}{2}\left(\boldsymbol{\Upsilon} + \boldsymbol{\Upsilon}^T\right) \\
\frac{\partial^2}{\partial \operatorname{vec}\boldsymbol{\Upsilon}\ \partial \boldsymbol{\mu}} \log p(\mathbf{y}) &= \frac{1}{2}\left(\mathbf{I}_d \otimes (\mathbf{y} - \boldsymbol{\mu}) + (\mathbf{y} - \boldsymbol{\mu}) \otimes \mathbf{I}_d\right) \\
&= \frac{1}{2}(\mathbf{I}_{d^2} + \mathbf{K}_d)\left(\mathbf{I}_d \otimes (\mathbf{y} - \boldsymbol{\mu})\right) \\
\frac{\partial^2}{\partial \operatorname{vec}\mathbf{F}\ \partial \boldsymbol{\mu}} \log p(\mathbf{y}) &= \mathbf{F}^T(\mathbf{y} - \boldsymbol{\mu}) \otimes \mathbf{I}_d + \mathbf{F}^T \otimes (\mathbf{y} - \boldsymbol{\mu}) \\
&= (\mathbf{F}^T \otimes \mathbf{I}_d)(\mathbf{I}_{d^2} + \mathbf{K}_d)\left(\mathbf{I}_d \otimes (\mathbf{y} - \boldsymbol{\mu})\right)
\end{aligned}$$

As in the case of natural parameter, we have

$$\begin{aligned}
d\,\frac{\partial}{\partial \boldsymbol{\Upsilon}} \log p(\mathbf{y}) &= -\frac{1}{2}\boldsymbol{\Upsilon}^{-T}(d\,\boldsymbol{\Upsilon}^T)\boldsymbol{\Upsilon}^{-T} \\
\frac{\partial^2}{\partial(\operatorname{vec}\boldsymbol{\Upsilon})^2} \log p(\mathbf{y}) &= -\frac{1}{2}\left(\boldsymbol{\Upsilon}^{-1} \otimes \boldsymbol{\Upsilon}^{-T}\right)\mathbf{K}_d \\
d\,\frac{\partial}{\partial \mathbf{F}} \log p(\mathbf{y}) &= -(\mathbf{y} - \boldsymbol{\mu})(\mathbf{y} - \boldsymbol{\mu})^T(d\,\mathbf{F}) - \mathbf{F}^{-T}(d\,\mathbf{F}^T)\mathbf{F}^{-T} \\
\frac{\partial^2}{\partial(\operatorname{vec}\mathbf{F})^2} \log p(\mathbf{y}) &= -\mathbf{I}_d \otimes (\mathbf{y} - \boldsymbol{\mu})(\mathbf{y} - \boldsymbol{\mu})^T - (\mathbf{F}^{-1} \otimes \mathbf{F}^{-T})\mathbf{K}_d
\end{aligned}$$

If we assume both $\boldsymbol{\Upsilon}$ and $d\,\boldsymbol{\Upsilon}$ are always symmetric, we have

$$\frac{\partial^2}{\partial(\operatorname{vec}\boldsymbol{\Upsilon})^2}\log p(\mathbf{y}) = -\frac{1}{2}\boldsymbol{\Sigma}\otimes\boldsymbol{\Sigma}$$

Bibliography

Bibliography

[1] S. Agarwal, J. Lim, L. Zelnik-Manor, P. Perona, D. Kriegman, and S. Belongie. Beyond pairwise clustering. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages II–838–II–845. IEEE Computer Society, 2005.

[2] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. 1998 ACM SIGMOD International Conference on Management of Data*, pages 94–105, June 1998.

[3] P. Arabie and L. Hubert. Cluster analysis in marketing research. In *Advanced Methods of Marketing Research*, pages 160–189. Oxford: Blackwell, 1994.

[4] L.D. Baker and A.K. McCallum. Distributional clustering of words for text classification. In *Proc. SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pages 96–103. ACM Press, New York, US, 1998.

[5] G.H. Bakir, J. Weston, and B. Schoelkopf. Learning to find pre-images. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[6] P. Baldi and G.W. Hatfield. *DNA Microarrays and Gene Expression*. Cambridge University Press, 2002.

[7] P. Baldi and K. Hornik. Neural networks and principal component analysis: learning from examples without local minima. *Neural Networks*, 2:53–58, 1989.

[8] G.H. Ball and D.J. Hall. Isodata, a novel method of data analysis and pattern classification. Technical Report NTIS AD 699616, Stanford Research Institute, Stanford, CA, 1965.

[9] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergence. In *Proc. SIAM International Conference on Data Mining*, pages 234–245, 2004.

[10] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[11] J. Bartram. A letter from John Bartram, M.D. to Peter Collinson, F.R.S. concerning a cluster of small teeth observed by him at the root of each fang or great tooth in the head of a rattle-snake, upon dissecting it. *Philosophical Transactions (1683–1775)*, 41:358–359, 1739–1741.

[12] S. Basu, A. Banerjee, and R.J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proc. the SIAM International Conference on Data Mining*, pages 333–344, 2004.

[13] S. Basu, A. Banerjee, and R.J. Mooney. Semi-supervised clustering by seeding. In *Proc. 19th International Conference on Machine Learning*, pages 19–26, 2005.

[14] S. Basu, M. Bilenko, and R.J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proc. 10th ACM SIGKDD, International Conference on Knowledge Discovery and Data Mining*, pages 59–68, 2004.

[15] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks*, 5(4):537–550, July 1994.

[16] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, June 2003.

[17] Y. Bengio, J.-F. Paiement, and P. Vincent. Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps, and spectral clustering. In *Advances in Neural Information Processing Systems 16*, pages 177–184. MIT Press, 2004.

[18] M. Bernstein, V. de Silva, J. Langford, and J. Tenenbaum. Graph approximations to geodesics on embedded manifolds. Technical report, Department of Psychology, Stanford University, 2000.

[19] A. Beygelzimer, S.M. Kakade, and J. Langford. Cover trees for nearest neighbor. Technical report, 2005. `http://www.cis.upenn.edu/~skakade/papers/ml/cover_tree.pdf`.

[20] S.K. Bhatia and J.S. Deogun. Conceptual clustering in information retrieval. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 28(3):427–436, June 1998.

[21] M. Bilenko, S. Basu, and R.J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proc. 21st International Conference on Machine Learning*, 2004. `http://doi.acm.org/10.1145/1015330.1015360`.

[22] J. Bins and B. Draper. Feature selection from huge feature sets. In *Proc. 8th IEEE International Conference on Computer Vision*, pages 159–165, 2001.

[23] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, New York, 1995.

[24] C.M. Bishop, M. Svensen, and C.K.I. Williams. GTM: the generative topographic mapping. *Neural Computation*, 10:215–234, 1998.

[25] G. Biswas, R. Dubes, and A.K. Jain. Evaluation of projection algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3:702–708, 1981.

[26] A.L. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

[27] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[28] P. Bradley, U. Fayyad, and C. Reina. Clustering very large database using EM mixture models. In *Proc. 15th International Conference on Pattern Recognition*, pages 76–80, September 2000.

[29] M. Brand. Charting a manifold. In *Advances in Neural Information Processing Systems 15*, pages 961–968. MIT Press, 2003.

[30] M. Brand. Fast online SVD revisions for lightweight recommender systems. In *Proc. SIAM International Conference on Data Mining*, 2003. `http://www.siam.org/meetings/sdm03/proceedings/sdm03_04.pdf`.

[31] M. Brand. Nonlinear dimensionality reduction by kernel eigenmaps. In *Proc. 18th International Joint Conference on Artificial Intelligence*, pages 547–552, August 2003.

[32] A. Brun, H-J. Park, H. Knutsson, and Carl-Fredrik Westin. Coloring of DT-MRI fiber traces using Laplacian eigenmaps. In *Proc. the Ninth International Conference on Computer Aided Systems Theory*, volume 2809, February 2003.

[33] J. Bruske and G. Sommer. Intrinsic dimensionality estimation with optimally topology preserving maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):572–575, 1998.

[34] C.J.C. Burges. Geometric methods for feature extraction and dimensional reduction. In L. Rokach and O. Maimon, editors, *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*. Kluwer Academic Publishers, 2005.

[35] F. Camastra and A. Vinciarelli. Estimating the intrinsic dimension of data with a fractal-based method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(10):1404–1407, October 2002.

[36] R. Caruana and D. Freitag. Greedy attribute selection. In *Proc. 11th International Conference on Machine Learning*, pages 28–36. Morgan Kaufmann, 1994.

[37] G. Celeux, S. Chrétien, F. Forbes, and A. Mkhadri. A component-wise EM algorithm for mixtures. *Journal of Computational and Graphical Statistics*, 10:699–712, 2001.

[38] Y. Chang, C. Hu, and M. Matthew Turk. Probabilistic expression analysis on manifolds. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 520–527, 2004.

[39] A. Chaturvedi and J.D. Carroll. A feature-based approach to market segmentation via overlapping k-centroids clustering. *Journal of Marketing Research*, 34(3):370–377, August 1997.

[40] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:790–799, 1995.

[41] Y. Cheng and G.M. Church. Biclustering of expression data. In *Proc. of the Eighth International Conference on Intelligent Systems for Molecular Biology*, 2000.

[42] F. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[43] Forrest E. Clements. Use of cluster analysis with anthropological data. *American Anthropologist, New Series, Part 1*, 56(2):180–199, April 1954.

[44] D. Comaniciu. An algorithm for data-driven bandwidth selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):281–288, 2003.

[45] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[46] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[47] J. Costa and A.O. Hero. Manifold learning using euclidean k-nearest neighbor graphs. In *Proc. IEEE International Conference on Acoustic Speech and Signal Processing*, volume 3, pages 988–991, Montreal, 2004.

[48] D.R. Cox. Note on grouping. *Journal of the American Statistical Association*, 52(280):543–547, December 1957.

[49] T.F. Cox and M.A.A. Cox. *Multidimensional Scaling*. Chapman & Hall, 2001.

[50] M. Craven, D. DiPasquo, D. Freitag, A.K. McCallum, T.M. Mitchell, K. Nigam, and S. Slattery. Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence*, 118(1/2):69–113, 2000.

[51] M. Dash and H. Liu. Feature selection for clustering. In *Proc. Pacific-Asia Conference on Knowledge Discovery and Data Mining 2000*, 2000.

[52] A. d'Aspremont, L. El Ghaoui, M.I. Jordan, and G.R.G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. In *Advances in Neural Information Processing Systems 17*. MIT Press, 2005.

[53] D. de Ridder, O. Kouropteva, O. Okun, M. Pietikinen, and R.P.W. Duin. Supervised locally linear embedding. In *Proc. Artificial Neural Networks and Neural Information Processing*, pages 333–341. Springer, 2003.

[54] D. de Ridder, M. Loog, and M.J.T. Reinders. Local fisher embedding. In *Proc. 17th International Conference on Pattern Recognition*, pages II–295–II–298, 2004.

[55] V. de Silva and J.B. Tenenbaum. Global versus local approaches to nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems 15*, pages 705–712. MIT Press, 2003.

[56] D. DeCoste. Visualizing Mercer kernel feature spaces via kernelized locally-linear embeddings. In *Proc. 8th International Conference on Neural Information Processing*, November 2001. Available at `http://www.cse.cuhk.edu.hk/~apnna/proceedings/iconip2001/index.htm`.

[57] D. DeMers and G. Cottrell. Non-linear dimensionality reduction. In *Advances in Neural Information Processing Systems 5*, pages 580–587. Morgan Kaufmann, 1993.

[58] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1–38, 1977.

[59] M. Dettling and P. Bühlmann. Finding predictive gene groups from microarray data. *Journal of Multivariate Analysis*, 90:106–131, 2004.

[60] M. Devaney and A. Ram. Efficient feature selection in conceptual clustering. In *Proc. 14th International Conference on Machine Learning*, pages 92–97. Morgan Kaufmann, 1997.

[61] I.S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Knowledge Discovery and Data Mining*, pages 269–274, 2001.

[62] I.S. Dhillon, S. Mallela, and R. Kumar. A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research*, 3:1265–1287, March 2003.

[63] I.S. Dhillon, S. Mallela, and D.S. Modha. Information-theoretic co-clustering. In *Proc. of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 89–98, 2003.

[64] D. Donoho. For most large underdetermined systems of linear equations, the minimal l^1-norm near-solution approximates the sparsest near-solution. Technical report, Department of Statistics, Stanford University, 2004.

[65] D. Donoho. For most large underdetermined systems of linear equations, the minimal l^1-norm solution is also the sparsest solution. Technical report, Department of Statistics, Stanford University, 2004.

[66] D.L. Donoho and C. Grimes. When does isomap recover natural parameterization of families of articulated images? Technical Report 2002-27, Department of Statistics, Stanford University, August 2002.

[67] D.L. Donoho and C. Grimes. Hessian eigenmaps: new locally linear embedding techniques for high-dimensional data. Technical Report TR-2003-08, Department of Statistics, Stanford University, 2003.

[68] R. Duda, P. Hart, and D. Stork. *Pattern Classification.* John Wiley & Sons, New York, 2nd edition, 2001.

[69] J.G. Dy and C.E. Brodley. Feature subset selection and order identification for unsupervised learning. In *Proc. 17th International Conference on Machine Learning*, pages 247–254. Morgan Kaufmann, 2000.

[70] J.G. Dy and C.E. Brodley. Feature selection for unsupervised learning. *Journal of Machine Learning Research*, 5:845–889, August 2004.

[71] J.G. Dy, C.E. Brodley, A. Kak, L.S. Broderick, and A.M. Aisen. Unsupervised feature selection applied to content-based retrieval of lung images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(3):373–378, March 2003.

[72] B. Efron, T. Hastie, I.M. Johnstone, and R. Tibshirani. Least angle regression (with discussion). *Annals of Statistics*, 32(2):407–499, 2004.

[73] R. El-Yaniv and O. Souroujon. Iterative double clustering for unsupervised and semi-supervised learning. In *Advances in Neural Information Processing Systems 14*, pages 1025–1032. MIT Press, 2002.

[74] A. Elgammal and C.S. Lee. Inferring 3D body pose from silhouettes using activity manifold learning. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 681–688, 2004.

[75] A. Elgammal and C.S. Lee. Separating style and content on a nonlinear manifold. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 478–489, 2004.

[76] D.M. Endres and J.E. Schindelin. A new metric for probability distributions. *IEEE Transactions on Information Theory*, 49:1858–1860, September 2003.

[77] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[78] M. Farmer and A. Jain. Occupant classification system for automotive airbag suppression. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 756–761, 2003.

[79] D. Fasulo. An analysis of recent work on clustering algorithms. Technical report, University of Washington, 1999. Available at `http://www.cs.washington.edu/homes/dfasulo/clustering.ps` and `http://citeseer.ist.psu.edu/fasulo99analysi.html`.

[80] M. Figueiredo. Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1150–1159, 2003.

[81] M.A.T. Figueiredo and A.K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

[82] M.A.T. Figueiredo, A.K. Jain, and M.H. Law. A feature selection wrapper for mixtures. In *Proc. the First Iberian Conference on Pattern Recognition and Image Analysis*, pages 229–237, Puerto de Andratx, Spain, 2003. Springer Verlag, LNCS vol. 2652.

[83] B. Fischer and J.M. Buhmann. Bagging for path-based clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1411–1415, 2003.

[84] B. Fischer and J.M. Buhmann. Path-based clustering for grouping smooth curves and texture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):513–518, 2003.

[85] W.D. Fisher. On grouping for maximum homogeneity. *Journal of the American Statistical Association*, 53(284):789–798, December 1958.

[86] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 1987.

[87] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 2nd edition, 2000.

[88] E. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classification. *Biometrics*, 21, September 1964.

[89] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):214–225, February 2004.

[90] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996.

[91] J.H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19(1):1–67, March 1991.

[92] J.H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823, 1981.

[93] J.H. Friedman and J.W. Tukey. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, C-23:881–890, 1974.

[94] H. Frigui and R. Krishnapuram. A robust competitive clustering algorithm with applications in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):450–465, May 1999.

[95] G. Getz, E. Levine, and E. Domany. Coupled two-way clustering analysis of gene microarray data. *Proc. National Academy of Sciences of the United States of America*, 94:12079–12084, 2000.

[96] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

[97] D. Gondek and T. Hofmann. Non-redundant data clustering. In *Proc. 5th IEEE International Conference on Data Mining*, pages 75–82, 2004.

[98] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems 17*, pages 529–536, Cambridge, MA, 2005. MIT Press.

[99] R.M. Gray and D.L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, October 1998.

[100] P. Gustafson, P. Carbonetto, N. Thompson, and N. de Freitas. Bayesian feature weighting for unsupervised learning, with application to object recognition. In C. M. Bishop and B. J. Frey, editors, *Proc. 9th International Workshop on Artificial Intelligence and Statistics*, Key West, FL, 2003.

[101] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, March 2003.

[102] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the NIPS 2003 feature selection challenge. In *Advances in Neural Information Processing Systems 17*, pages 545–552. MIT Press, 2005.

[103] A. Hadid, O. Kouropteva, and M. Pietikainen. Unsupervised learning using locally linear embedding: experiments in face pose analysis. In *Proc. 16th International Conference on Pattern Recognition*, pages I:111–114, 2002.

[104] M.A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proc. 17th International Conference on Machine Learning*, pages 359–366. Morgan Kaufmann, 2000.

[105] J. Ham, D.D. Lee, S. Mika, and B. Schoelkopf. A kernel view of the dimensionality reduction of manifolds. In *Proc. 21st International Conference on Machine Learning*, 2004.

[106] G. Hamerly and C. Elkan. Learning the k in k-means. In *Advances in Neural Information Processing Systems 16*, pages 281–288. MIT Press, 2004.

[107] T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84:502–516, 1989.

[108] T. Hastie, R. Tibshirani, D. Botstein, and P. Brown. Supervised harvesting of expression trees. *Genome Biology*, 2:0003.1–0003.12, 2003.

[109] X. He and P. Niyogi. Locality preserving projections. In *Advances in Neural Information Processing Systems 16*, pages 153–160. MIT Press, 2004.

[110] T. Hertz, N. Shental, A. Bar-Hillel, and D. Weinshall. Enhancing image and video retrieval: Learning via equivalence constraint. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 668–674, 2003.

[111] A. Hinneburg and D.A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Knowledge Discovery and Data Mining*, pages 58–65, 1998.

[112] G. Hinton and S. Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems 15*, pages 833–840. MIT Press, 2003.

[113] G.E. Hinton, P. Dayan, and M. Revow. Modeling the manifolds of handwritten digits. *IEEE Transactions on Neural Networks*, 8(1):65–74, January 1997.

[114] T. Hofmann and J. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):1–14, 1997.

[115] T. Hofmann and J.M. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):1–14, 1997.

[116] T. Hofmann, J. Puzicha, and J.M. Buhmann. Unsupervised texture segmentation in a deterministic annealing framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):803–818, 1998.

[117] R. Horst, P.M. Pardalos, and Nguyen Van Thoai. *Introduction to Global Optimization (Nonconvex Optimization and Its Applications)*. Springer, 1995.

[118] W.S. Hwang and J. Weng. Hierarchical discriminant regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1277–1293, November 2000.

[119] P. Indyk and J. Matousek. Low distortion embeddings of finite metric spaces. In *Handbook of Discrete and Computational Geometry*, pages 177–196. CRC Press LLC, 2nd edition, 2004. `http://theory.lcs.mit.edu/~indyk/p.html`.

[120] M. Iwayama and T. Tokunaga. Cluster-based text categorization: a comparison of category search strategies. In *Proc. of 18th ACM International Conference on Research and Development in Information Retrieval*, pages 273–281, 1995.

[121] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*, pages 487–493, 1998.

[122] A. Jain and D. Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–157, February 1997.

[123] A.K. Jain, S.C. Dass, and K. Nandakumar. Soft biometric traits for personal recognition systems. In *Proc. International Conference on Biometric Authentication*, pages 731–738, Hong Kong, 2004.

[124] A.K. Jain and R. Dubes. Feature definition in pattern recognition with small sample size. *Pattern Recognition*, 10(2):85–97, 1978.

[125] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[126] A.K. Jain, R. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–38, 2000.

[127] A.K. Jain and F. Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern Recognition*, 24:1167–1186, 1991.

[128] A.K. Jain and P. Flynn. Image segmentation using clustering. In *Advances in Image Understanding*, pages 65–83. IEEE Computer Society Press, 1996.

[129] A.K. Jain and J. Mao. Artificial neural network for nonlinear projection of multivariate data. In *Proc. International Joint Conference on Neural Networks*, pages III–335–III–340, 1992.

[130] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[131] A.K. Jain, A. Topchy, M.H. Law, and J. Buhmann. Landscape of clustering algorithms. In *Proc. 17th International Conference on Pattern Recognition*, pages I–260–I–263, 2004.

[132] A.K. Jain and D. Zongker. Representation and recognition of handwritten digits using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1386–1390, December 1997.

[133] O. Jenkins and M. Mataric. A spatio-temporal extension to Isomap nonlinear dimension reduction. In *Proc. 21st International Conference on Machine Learning*, 2004. Available at `http://doi.acm.org/10.1145/1015330.1015357`.

[134] S.T. Roweis J.J. Verbeek and N. Vlassis. Non-linear CCA and PCA by alignment of local models. In *Advances in Neural Information Processing Systems 16*, pages 297–304. MIT Press, 2004.

[135] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proc. 10th European Conference on Machine Learning*, pages 137–142. Springer Verlag, 1998.

[136] T. Joachims. Transductive inference for text classification using support vector machines. In *Proc. 16th International Conference on Machine Learning*, pages 200–209. Morgan Kaufmann, 1999.

[137] I.M. Johnstone and A.Y. Lu. Sparse principal components analysis. Located at `http://www-stat.stanford.edu/~imj/WEBLIST/AsYetUnpub/sparse.pdf`, 2004.

[138] S. Kamvar, D. Klein, and C.D. Manning. Spectral learning. In *Proc. 18th International Joint Conference on Artificial Intelligence*, pages 561–566, 2003.

[139] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM*, 51(3):497–515, May 2004.

[140] T. Kanungo, D.M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.

[141] J.N. Kapur. *Measures of Information and Their Applications*. Wiley, New Delhi, India, 1994.

[142] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithms using dynamic modeling. *IEEE Computer, Special Issue on Data Analysis and Mining*, 32(8):68–75, August 1999.

[143] B. Kégl. Intrinsic dimension estimation using packing numbers. In *Advances in Neural Information Processing Systems 15*, pages 681–688. MIT Press, 2003.

[144] B. Kégl, A. Krzyzak, T. Linder, and K. Zeger. Learning and design of principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(3):281–297, 2000.

[145] W.-Y. Kim and A.C. Kak. 3-d object recognition using bipartite matching embedded in discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):224–251, March 1991.

[146] Y. Kim, W. Street, and F. Menczer. Feature selection in unsupervised learning via evolutionary search. In *Proc. 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 365–369, 2000.

[147] K. Kira and L. Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Proc. of the 10th National Conference on Artificial Intelligence*, pages 129–134, Menlo Park, CA, USA, 1992. AAAI Press.

[148] D. Klein, S.D. Kamvar, and C.D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *Proc. 19th International Conference on Machine Learning*, pages 307–314. Morgan Kaufmann, 2002.

[149] J. Kleinberg. An impossibility theorem for clustering. In *Advances in Neural Information Processing Systems 15*. MIT Press, 2003.

[150] Y. Kluger, R. Basri, J.T. Chang, and M. Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Research*, 13:703–716, 2003.

[151] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[152] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, 2001. 3rd edition.

[153] D. Koller and M. Sahami. Toward optimal feature selection. In *Proc. 13th International Conference on Machine Learning*, pages 284–292. Morgan Kaufmann, 1996.

[154] R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proc. 19th International Conference on Machine Learning*, pages 315–322. Morgan Kaufmann, 2002.

[155] I. Kononenko. Estimating attributes: Analysis and extensions of RELIEF. In *Proc. 7th European Conference on Machine Learning*, pages 171–182, 1994.

[156] B. Krishnapuram, A. Hartemink, L. Carin, and M. Figueiredo. A Bayesian approach to joint feature selection and classifier design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1105–1111, September 2004.

[157] B. Krishnapuram, D. Williams, Y. Xue, A. Hartemink, L. Carin, and M. Figueiredo. On semi-supervised classification. In *Advances in Neural Information Processing Systems 17*, pages 721–728, Cambridge, MA, 2005. MIT Press.

[158] B. Kulis, S. Basu, I. Dhillon, and R. Mooney. Semi-supervised graph clustering: A kernel approach. In *Proc. 22nd International Conference on Machine Learning*, pages 457–464, 2005.

[159] N. Kwak and C.-H. Choi. Input feature selection by mutual information based on Parzen window. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1667–1671, December 2002.

[160] J.T. Kwok and I.W. Tsang. The pre-image problem in kernel methods. In *Proc. 20th International Conference on Machine Learning*, pages 408–415, 2003.

[161] T. Lange, M.H. Law, A.K. Jain, and J.B. Buhmann. Learning with constrained and unlabelled data. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 730–737, 2005.

[162] T. Lange, V. Roth, M.L. Braun, and J.M. Buhmann. Stability-based validation of clustering solutions. *Neural Computation*, 16(6):1299–1323, June 2004.

[163] M.H. Law, M.A.T. Figueiredo, and A.K. Jain. Simultaneous feature selection and clustering using mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1154–1166, September 2004.

[164] M.H. Law and A.K. Jain. Incremental nonlinear dimensionality reduction by manifold learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):377–391, March 2006.

[165] M.H. Law, A.K. Jain, and M.A.T. Figueiredo. Feature selection in mixture-based clustering. In *Advances in Neural Information Processing Systems 15*, pages 625–632. MIT Press, 2003.

[166] M.H. Law, A. Topchy, and A.K. Jain. Clustering with soft and group constraints. In *Proc. Joint IAPR International Workshops on Structural, Syntactic, And Statistical Pattern Recognition*, pages 662–670, Lisbon, Portugal, August 2004.

[167] M.H. Law, A. Topchy, and A.K. Jain. Model-based clustering with probabilistic constraints. In *Proc. SIAM International Conference on Data Mining*, pages 641–645, 2005.

[168] M.H. Law, N. Zhang, and A.K. Jain. Non-linear manifold learning for data stream. In *Proc. SIAM International Conference for Data Mining*, pages 33–44, 2004.

[169] N.D. Lawrence and M.I. Jordan. Semi-supervised learning via gaussian processes. In *Advances in Neural Information Processing Systems 17*, pages 753–760, Cambridge, MA, 2005. MIT Press.

[170] Y. LeCun, O. Matan, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, and H.S. Baird. Handwritten zip code recognition with multilayer networks. In *Proc. 10th International Conference on Pattern Recognition*, volume 2, pages 35–40, 1990.

[171] E. Levina and P.J. Bickel. Maximum likelihood estimation of intrinsic dimension. In *Advances in Neural Information Processing Systems 17*, pages 777–784. MIT Press, 2005.

[172] S.Z. Li, R. Xiao, Z.Y. Li, and H.J. Zhang. Nonlinear mapping from multi-view face patterns to a gaussian distribution in a low dimensional space. In *Proc. IEEE ICCV Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems*, 2001. Available at `http://doi.ieeecomputersociety.org/10.1109/RATFG.2001.938909`.

[173] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

[174] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982. Originally as an unpublished Bell Laboratories Technical Note (1957).

[175] X. Lu and A.K. Jain. Ethnicity identification from face images. In *Proc. SPIE*, volume 5404, pages 114–123, 2004.

[176] Z. Lu and T. Leen. Semi-supervised learning with penalized probabilistic clustering. In *Advances in Neural Information Processing Systems 17*, pages 849–856, Cambridge, MA, 2005. MIT Press.

[177] D.J.C. MacKay. Bayesian non-linear modelling for the prediction competition. In *ASHRAE Transactions, V.100, Pt.2*, pages 1053–1062, Atlanta Georgia, 1994.

[178] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symposium on Math. Stat. and Prob.*, pages 281–297. University of California Press, 1967.

[179] J.R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiles and Sons, 1999. Revised Edition.

[180] J. Mao and A.K. Jain. Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, 6(2):296–317, March 1995.

[181] A. Martinez and R. Benavente. The AR face database. Technical Report 24, CVC, 1998. `http://rvl1.ecn.purdue.edu/~aleix/aleix_face_DB.html`.

[182] M.Belkin and P. Niyogi. Semi-supervised learning on manifolds. *Machine Learning, Special Issue on Clustering*, 56:209–239, 2004.

[183] G. McLachlan and K. Basford. *Mixture Models: Inference and Application to Clustering*. Marcel Dekker, New York, 1988.

[184] G. McLachlan and D. Peel. *Finite Mixture Models*. John Wiley & Sons, New York, 2000.

[185] S. Mika, B. Scholkopf, A.J. Smola, K.-R. Muller, M. Scholz, and G. Ratsch. Kernel PCA and de-noising in feature spaces. In *Advances in Neural Information Processing Systems 11*, pages 536–542. MIT Press, 1998.

[186] A.J. Miller. *Subset Selection in Regression*. Chapman & Hall, London, 2002.

[187] B. Mirkin. Concept learning and feature selection based on square-error clustering. *Machine Learning*, 35:25–39, 1999.

[188] P. Mitra and C.A. Murthy. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):301–312, 2002.

[189] K.-R. Mller, S. Mika, G. Rtsch, K. Tsuda, and B. Schlkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, May 2001.

[190] D. Modha and W. Scott-Spangler. Feature weighting in k-means clustering. *Machine Learning*, 52(3):217–237, September 2003.

[191] F. Mosteller and J.W. Tukey. *Data Analysis and Regression*. Addison-Wesley, Boston, 1977.

[192] A.M. Namboodiri and A.K. Jain. Online handwritten script recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):124–130, 2004.

[193] P. Narváez, K.-Y. Siu, and H.-Y. Tzeng. New dynamic algorithms for shortest path tree computation. *IEEE/ACM Transactions on Networking*, 8(6):734–746, December 2000.

[194] A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856, Cambridge, MA, 2002. MIT Press.

[195] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[196] M. Niskanen and O. Silvn. Comparison of dimensionality reduction methods for wood surface inspection. In *Proc. 6th International Conference on Quality Control by Artificial Vision*, pages 178–188, 2003.

[197] J. Novovicová, P. Pudil, and J. Kittler. Divergence based feature selection for multimodal class densities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):218–223, February 1996.

[198] M.R. Osborne, B. Presnell, and B.A. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20(3):389–403, 2000.

[199] F. Österreicher and I. Vajda. A new class of metric divergences on probability spaces and its statistical applications. *Ann. Inst. Statist. Math.*, 55:639–653, 2003.

[200] E.M. Palmer. *Graph Evolution: An Introduction to the Theory of Random Graphs*. John Wiley & Sons, 1985.

[201] A. Patrikainen and H. Mannila. Subspace clustering of high-dimensional binary data – a probabilistic approach. In *Proc. Workshop on Clustering High Dimensional Data, SIAM International Conference on Data Mining*, pages 57–65, 2004.

[202] E. Pekalska, P. Paclik, and R.P.W. Duin. A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research*, 2:175–211, December 2001.

[203] D. Pelleg and A.W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proc. 17th International Conference on Machine Learning*, pages 727–734, San Francisco, 2000. Morgan Kaufmann.

[204] J. Pena, J. Lozano, P. Larranaga, and I. Inza. Dimensionality reduction in unsupervised learning of conditional Gaussian networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):590–603, 2001.

[205] W. Penny and S. Roberts. Notes on variational learning. Technical Report PARG-00-1, Department of Engineering Science, University of Oxford, April 2000.

[206] P. Perona and M. Polito. Grouping and dimensionality reduction by locally linear embedding. In *Advances in Neural Information Processing Systems 14*, pages 1255–1262. MIT Press, 2002.

[207] K. Pettis, T. Bailey, A.K. Jain, and R. Dubes. An intrinsic dimensionality estimator from near-neighbor information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(1):25–36, January 1979.

[208] K.S. Pollard and M.J. van der Laan. Statistical inference for simultaneous clustering of gene expression data. *Mathematical Biosciences*, 176(1):99–121, 2002.

[209] P. Pudil, J. Novovicovà, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15:1119–1125, 1994.

[210] P. Pudil, J. Novovicová, and J. Kittler. Feature selection based on the approximation of class densities by finite mixtures of the special type. *Pattern Recognition*, 28(9):1389–1398, 1995.

[211] S.J. Raudys and A.K. Jain. Small sample size effects in statistical pattern recognition: Recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):252–264, 1991.

[212] M.L. Raymer, W.F. Punch, E.D. Goodman, L.A. Kuhn, and A.K. Jain. Dimensionality reduction using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(2):164–171, July 2000.

[213] T. Sørensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons. *Biologiske Skrifter*, 5:1–34, 1948.

[214] D. De Ridder and V. Franc. Robust manifold learning. Technical Report CTU-CMP-2003-08, Center for Machine Perception, Department of Cybernetics Faculty of Electrical Engineering, Czech Technical University, Prague, 2003.

[215] J. Rissanen. *Stochastic Complexity in Stastistical Inquiry.* World Scientific, Singapore, 1989.

[216] S.J. Roberts, R.M. Everson, and I. Rezek. Maximum certainty data partitioning. *Pattern Recognition*, 33(5):833–839, 1999.

[217] A. Robles-Kelly and E.R. Hancock. A graph spectral approach to shape-from-shading. *IEEE Transactions on Image Processing*, 13:912–926, 2004.

[218] V. Roth and T. Lange. Feature selection in clustering problems. In *Advances in Neural Information Processing Systems 16.* MIT Press, Cambridge, MA, 2004.

[219] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[220] S.T. Roweis, L.K. Saul, and G.E. Hinton. Global coordination of local linear models. In *Advances in Neural Information Processing Systems 14*, pages 889–896. MIT Press, 2002.

[221] M. Sahami. *Using Machine Learning to Improve Information Access.* PhD thesis, Computer Science Department, Stanford University, 1998.

[222] R. Salakhutdinov, S.T. Roweis, and Z. Ghahramani. Optimization with em and expectation-conjugate-gradient. In *Proc. 20th International Conference on Machine Learning*, pages 672–679. AAAI Press, 2003.

[223] J.W. Sammon. A non-linear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, May 1969.

[224] P. Sand and A.W. Moore. Repairing faulty mixture models using density estimation. In *Proc. 18th International Conference on Machine Learning*, pages 457–464. Morgan Kaufmann, 2001.

[225] W.S. Sarle. The VARCLUS procedure. In *SAS/STAT User's Guide*. SAS Institute, Inc., 4th edition, 1990.

[226] L.K. Saul and S.T. Roweis. Think globally, fit locally: Unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[227] C. Saunders, J. Shawe-Taylor, and A. Vinokourov. String kernels, fisher kernels and finite state automata. In *Advances in Neural Information Processing Systems 15*, pages 633–640. MIT Press, 2003.

[228] B. Schölkopf, A.J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[229] B. Schölkopf, K. Sung, C.J.C. Burges, F. Girosi, P. Niyogi, T. Poggio, and V. Vapnik. Comparing support vector machines with gaussian kernels to radial basis function classifiers. *IEEE Transactions on Signal Processing*, 45(11):2758–2765, 1997.

[230] M. Segal, K. Dahlquist, and B. Conklin. Regression approach for microarray data analysis. *Journal of Computational Biology*, 10:961–980, 2003.

[231] N. Shental, A. Bar-Hillel, T. Hertz, and D. Weinshall. Computing gaussian mixture models with EM using equivalence constraints. In *Advances in Neural Information Processing Systems 16*, pages 465–472. MIT Press, 2004.

[232] J.R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, School of Computer Science, Carnegie Mellon University, August 1994. Available at `http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf`.

[233] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 731–737, 1997.

[234] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[235] P.Y. Simard, D. Steinkraus, and J. Platt. Best practice for convolutional neural networks applied to visual document analysis. In *Proc. International Conference on Document Analysis and Recogntion*, pages 958–962, 2003.

[236] E. Sjöström. Singular value computations for toeplitz matrices. *Licentiate thesis*, 1996. `http://www.mai.liu.se/~evlun/pub/lic/lic.html`.

[237] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. In *ACM SIGIR*, pages 208–215, 2000.

[238] N. Slonim and N. Tishby. The power of word clusters for text classification. In *23rd European Colloquium on Information Retrieval Research*, 2001.

[239] A.J. Smola, S. Mika, B. Schölkopf, and R.C. Williamson. Regularized principal manifolds. *Journal of Machine Learning Research*, 1:179–209, June 2001.

[240] P.H.A. Sneath. The application of computers to taxonomy. *J. Gen. Microbiol.*, 17:201–226, 1957.

[241] R.R. Sokal and C.D. Michener. A statistical method for evaluating systematic relationships. *Univ. Kansas Sci. Bull.*, 38:1409–1438, 1958.

[242] R.R. Sokal and P.H.A. Sneath. *Principles of Numerical Taxonomy.* San Francisco, W. H. Freeman, 1963.

[243] H. Steinhaus. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci., C1. III, Vol. IV*, pages 801–804, 1956.

[244] S.S. Stevens. On the theory of the scales of measurement. *Science*, 103:677–680, 1946.

[245] E. Sungur. Overview of multivariate statistical data analysis, `http://www.mrs.umn.edu/~sungurea/multivariatestatistics/overview.html`.

[246] L. Talavera. Dependency-based feature selection for clustering symbolic data. *Intelligent Data Analysis*, 4:19–28, 2000.

[247] Y.W. Teh and S.T. Roweis. Automatic alignment of local representations. In *Advances in Neural Information Processing Systems 15*, pages 841–848. MIT Press, 2003.

[248] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[249] R. Tibshirani. Principal curves revisited. *Statistics and Computing*, 2:183–190, 1992.

[250] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[251] K. Torkkola. Feature extraction by non parametric mutual information maximization. *Journal of Machine Learning Research*, 3:1415–1438, March 2003.

[252] G. Trunk. A problem of dimensionality: A simple example. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(3):306–307, 1979.

[253] M.A. Turk and A.P. Pentland. Face recognition using eigenfaces. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991.

[254] S. Vaithyanathan and B. Dom. Generalized model selection for unsupervised learning in high dimensions. In *Advances in Neural Information Processing Systems 12*, pages 970–976, Cambridge, MA, 1999. MIT Press.

[255] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 2nd edition, 2000.

[256] P.F. Velleman and L. Wilkinson. Nominal, ordinal, interval, and ratio typologies are misleading. *The American Statistician*, 47(1):65–72, February 1993.

[257] J.J. Verbeek, N. Vlassis, and B. Krose. Coordinating principal component analyzers. In *Proc. International Conference on Artificial Neural Networks*, pages 914–919, 2002.

[258] D. Verma and M. Meila. A comparison of spectral clustering algorithms. Technical Report 03-05-01, CSE Department, University of Washington, 2003.

[259] P. Verveer and R. Duin. An evaluation of intrinsic dimensionality estimators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):81–86, 1995.

[260] P. Vincent and Y. Bengio. Manifold parzen windows. In *Advances in Neural Information Processing Systems 15*, pages 825–832. MIT Press, 2003.

[261] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages I–511–I–518, vol. 1, 2001.

[262] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *Proc. 17th International Conference on Machine Learning*, pages 1103–1110. Morgan Kaufmann, 2000.

[263] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proc. 18th International Conference on Machine Learning*, pages 577–584. Morgan Kaufmann, 2001.

[264] C.S. Wallace and D.L. Dowe. MML clustering of multi-state, Poisson, von Mises circular and Gaussian distributions. *Statistics and Computing*, 10:73–83, 2000.

[265] C.S. Wallace and P. Freeman. Estimation and inference via compact coding. *Journal of the Royal Statistical Society. Series B (Methodological)*, 49(3):241–252, 1987.

[266] J. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, March 1963.

[267] K.Q. Weinberger, B. Packer, and L.K. Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *Proc. 10th International Workshop on Artificial Intelligence and Statistics*, pages 381–388, 2005.

[268] K.Q. Weinberger and L.K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 988–995, 2004.

[269] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *Proc. 7th IEEE International Conference on Computer Vision*, pages II–975–II–982, 1999.

[270] J. Weng, Y. Zhang, and W.S. Hwang. Candid covariance-free incremental principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):1034–1040, 2003.

[271] D. Wettschereck, D.W. Aha, and T. Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artif. Intell. Rev.*, 11(1-5):273–314, 1997.

[272] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, November 1993.

[273] E. Xing, M. Jordan, and R. Karp. Feature selection for high-dimensional genomic microarray data. In *Proc. 18th International Conference on Machine Learning*, pages 601–608. Morgan Kaufmann, 2001.

[274] E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, 2003.

[275] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems*, 13:44–49, 1998.

[276] M.-H. Yang. Face recognition using extended isomap. In *IEEE International Conference on Image Processing*, pages II: 117–120, 2002.

[277] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proc. SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, pages 42–49. ACM Press, New York, 1999.

[278] L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proc. 20th International Conference on Machine Learning*, pages 856–863. AAAI Press, 2003.

[279] S.X. Yu and J. Shi. Segmentation given partial grouping constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):173–183, 2004.

[280] C.T. Zahn. Graph-theoretic methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, 20(31):68–86, 1971.

[281] H. Zha, X. He, C. Ding, M. Gu, and H. Simon. Bipartite graph partitioning and data clustering. In *Proc. of ACM 10th Int'l Conf. Information and Knowledge Management*, pages 25–31, 2001.

[282] H. Zha and Z. Zhang. Isometric embedding and continuum isomap. In *Proc. 20th International Conference on Machine Learning*, pages 864–871, 2003.

[283] J. Zhang, Stan Z. Li, and J. Wang. Nearest manifold approach for face recognition. In *Proc. The 6th International Conference on Automatic Face and Gesture Recognition*, Seoul, Korea, May 2004. Available at `http://www.cbsr.ia.ac.cn/users/szli/papers/ZJP-FG2004.pdf`.

[284] J. Zhang, S.Z. Li, and J. Wang. Manifold learning and applications in recognition. In *Intelligent Multimedia Processing with Soft Computing*. Springer-Verlag, Heidelberg, 2004. Available at `http://www.nlpr.ia.ac.cn/users/szli/papers/ZHP-MLA-Chapter.pdf`.

[285] Z. Zhang and H. Zha. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. Technical Report CSE-02-019, CSE, Penn State Univ., 2002.

[286] Q. Zhao and D.J. Miller. Mixture modeling with pairwise, instance-level class constraints. *Neural Computation*, 17(11):2482–2507, November 2005.

[287] D. Zhou, B. Schoelkopf, and T. Hofmann. Semi-supervised learning on directed graphs. In *Advances in Neural Information Processing Systems 17*, pages 1633–1640, Cambridge, MA, 2005. MIT Press.

[288] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. 20th International Conference on Machine Learning*, pages 912–919. AAAI Press, 2003.

[289] X. Zhu, J. Kandola, Z. Ghahramani, and J. Lafferty. Nonparametric transforms of graph kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems 17*, pages 1641–1648, Cambridge, MA, 2005. MIT Press.

[290] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Methodological)*, To appear.

[291] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal components analysis. Technical report, Department of Statistics, Stanford University, 2004.