2009

Learning Techniques For Information Retrieval And Mining In High-dimensional Databases

Hao Cheng

University of Central Florida

Part of the Computer Sciences Commons, and the Engineering Commons

Find similar works at: https://stars.library.ucf.edu/etd

University of Central Florida Libraries http://library.ucf.edu

This Doctoral Dissertation (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Electronic Theses and Dissertations, 2004-2019 by an authorized administrator of STARS. For more information, please contact STARS@ucf.edu.

STARS Citation

LEARNING TECHNIQUES FOR INFORMATION RETRIEVAL AND MINING IN HIGH-DIMENSIONAL DATABASES

by

HAO CHENG
M.S., University of Central Florida, 2008
B.S., University of Science and Technology of China, 2004

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the School of Electrical Engineering and Computer Science in the College of Engineering and Computer Science at the University of Central Florida Orlando, Florida

Fall Term
2009

Major Professor: Kien A. Hua
ABSTRACT

The main focus of my research is to design effective learning techniques for information retrieval and mining in high-dimensional databases. There are two main aspects in the retrieval and mining research: accuracy and efficiency. The accuracy problem is how to return results which can better match the ground truth, and the efficiency problem is how to evaluate users’ requests and execute learning algorithms as fast as possible. However, these problems are non-trivial because of the complexity of the high-level semantic concepts, the heterogeneous natures of the feature space, the high dimensionality of data representations and the size of the databases. My dissertation is dedicated to addressing these issues. Specifically, my work has five main contributions as follows.

The first contribution is a novel manifold learning algorithm, *Local and Global Structures Preserving Projection (LGSPP)*, which defines salient low-dimensional representations for the high-dimensional data. A small number of projection directions are sought in order to properly preserve the local and global structures for the original data. Specifically, two groups of points are extracted for each individual point in the dataset: the first group contains the nearest neighbors of the point, and the other set are a few sampled points far away from the point. These two point sets respectively characterize the local and global structures with regard to the data point. The objective of the embedding is to minimize the distances of the points in each local neighborhood and also to disperse the points far away from their respective remote points in the original space. In this way, the relationships between the data in the original space are well preserved with little distortions.

The second contribution is a new constrained clustering algorithm. Conventionally, clustering is an unsupervised learning problem, which systematically partitions a dataset into a small set of clusters such that data in each cluster appear similar to each other compared with those in other clusters. In the proposal, the partial human knowledge is exploited to find better clustering results. Two kinds of constraints are integrated into the clustering
algorithm. One is the must-link constraint, indicating that the involved two points belong to the same cluster. On the other hand, the cannot-link constraint denotes that two points are not within the same cluster. Given the input constraints, data points are arranged into small groups and a graph is constructed to preserve the semantic relations between these groups. The assignment procedure makes a best effort to assign each group to a feasible cluster without violating the constraints. The theoretical analysis reveals that the probability of data points being assigned to the true clusters is much higher by the new proposal, compared to conventional methods. In general, the new scheme can produce clusters which can better match the ground truth and respect the semantic relations between points inferred from the constraints.

The third contribution is a unified framework for partition-based dimension reduction techniques, which allows efficient similarity retrieval in the high-dimensional data space. Recent similarity search techniques, such as Piecewise Aggregate Approximation (PAA), Segmented Means (SMEAN) and Mean-Standard deviation (MS), prove to be very effective in reducing data dimensionality by partitioning dimensions into subsets and extracting aggregate values from each dimension subset. These partition-based techniques have many advantages including very efficient multi-phased pruning while being simple to implement. They, however, are not adaptive to different characteristics of data in diverse applications. In this study, a unified framework for these partition-based techniques is proposed and the issue of dimension partitions is examined in this framework. An investigation of the relationships of query selectivity and the dimension partition schemes discovers indicators which can predict the performance of a partitioning setting. Accordingly, a greedy algorithm is designed to effectively determine a good partitioning of data dimensions so that the performance of the reduction technique is robust with regard to different datasets.

The fourth contribution is an effective similarity search technique in the database of point sets. In the conventional model, an object corresponds to a single vector. In the proposed study, an object is represented by a set of points. In general, this new representation can
be used in many real-world applications and carries much more local information, but the retrieval and learning problems become very challenging. The Hausdorff distance is the common distance function to measure the similarity between two point sets, however, this metric is sensitive to outliers in the data. To address this issue, a novel similarity function is defined to better capture the proximity of two objects, in which a one-to-one mapping is established between vectors of the two objects. The optimal mapping minimizes the sum of distances between each paired points. The overall distance of the optimal matching is robust and has high retrieval accuracy. The computation of the new distance function is formulated into the classical assignment problem. The lower-bounding techniques and early-stop mechanism are also proposed to significantly accelerate the expensive similarity search process.

The classification problem over the point-set data is called *Multiple Instance Learning* (MIL) in the machine learning community in which a vector is an instance and an object is a bag of instances. The fifth contribution is to convert the MIL problem into a standard supervised learning in the conventional vector space. Specially, feature vectors of bags are grouped into clusters. Each object is then denoted as a bag of cluster labels, and common patterns of each category are discovered, each of which is further reconstructed into a bag of features. Accordingly, a bag is effectively mapped into a feature space defined by the distances from this bag to all the derived patterns. The standard supervised learning algorithms can be applied to classify objects into pre-defined categories. The results demonstrate that the proposal has better classification accuracy compared to other state-of-the-art techniques.

In the future, I will continue to explore my research in large-scale data analysis algorithms, applications and system developments. Especially, I am interested in applications to analyze the massive volume of online data.
I would like to thank the University of Central Florida (UCF) for offering assistantships that supports my PhD study. The university has provided a comfortable and convenient environment for me to work on my dissertation research.

I would like to express my sincere gratitude to my advisor, Dr. Kien A. Hua. He is very busy, and always manages to spare his valuable time to discuss with me on various research and also social topics, which greatly opens my mind set. He always encourages me to work on promising research areas and motives me to concentrate on my graduate study. I have been given great trust and freedom to work on the problems of my interest. He is enthusiastic, keen and open to new ideas. Dr. Hua has taught me a lot. I acknowledge his patience, sincerity and all his helps and supports extended to me throughout the years.

It is really my honor to have Dr. Michael Georgiopoulos, Dr. Charles E. Hughes, Dr. Sheau-Dong Lang and Dr. Xin Li as my committee members for my dissertation research. I would like to thank them all for their insightful comments and feedbacks. I indeed appreciate their helps and supports for me to work on my dissertation.

I would like to thank my colleague, Dr. Khanh Vu. In 2005, I did the coding and experiments with him on the dimensionality reduction paper, later published in SIGMOD 2006. This is the first paper, having my name appear in an internationally renowned conference. This great experience motivated me to work in the field of similarity search and later choose this as the research direction in my dissertation. Since then, I have been working closely with Dr. Vu on several other papers. I really appreciate his discussions and great help in revising my papers.

Thanks should also go to all the members in the Data System Groups. We have been living and working together for years. It is very fortunate to have them as labmates. We work together on projects, discuss on various topics, and share resources with each other. I have learned a lot from them. This experience is indeed valuable.
Above all, I thank my family: my parents and my elder brother. They are always helping me with no reservations. They always teach me to be a good and active person. They always share their faithful opinions and guide me through my path of growing up. I also would like to thank my parents in law. They take good care of me. Especially, I would like to thank my wife, Xiaoxiao Zhou. She has been an essential part of my life since we are together. My life is full of joy with her. She has helped train me a better person. Her love, understanding and support keep me focus and working diligently toward my doctor degree.

Last but not least, I would also like to extend my thanks to those who have cared and helped, in one way or another, in making this dissertation research possible.
# TABLE OF CONTENTS

LIST OF FIGURES xiii

LIST OF TABLES xiv

LIST OF ACRONYMS xv

CHAPTER 1: INTRODUCTION 1

1.1 Publications ........................................... 1
1.2 Motivations and Contributions ........................ 3
1.3 Overview of the Dissertation .......................... 7

CHAPTER 2: LOCAL AND GLOBAL STRUCTURES PRESERVING PROJECTION 8

2.1 Introduction ............................................. 8
   2.1.1 Related Work ....................................... 8
   2.1.2 Motivation Of Our Approach ......................... 9
2.2 Local and Global Structures Extraction ................ 10
2.3 Optimal Projection ..................................... 14
   2.3.1 Problem Formulation ................................ 14
   2.3.2 Problem Solution .................................. 15
2.4 Experimental Results ................................... 17
   2.4.1 Synthetic Data Examples ........................... 17
   2.4.2 Clustering Evaluation .............................. 21
2.5 Summary .................................................. 24
2.6 Proofs .................................................... 25
CHAPTER 3: SUBSPACE CLUSTERING AND SEMI-SUPERVISED CLUSTERING

3.1 Introduction .............................................................. 28
3.2 Related Work ............................................................. 30
3.3 Locally Weighted Clustering ........................................... 33
3.4 Clustering Under Constraints ......................................... 37
   3.4.1 Chunklet Assignment Basics ................................... 37
   3.4.2 Constrained Clustering ......................................... 42
3.5 Experimental Results ................................................... 45
   3.5.1 Methods and Datasets ........................................... 45
   3.5.2 Evaluation Metrics .............................................. 47
   3.5.3 Unsupervised Clustering Accuracy .............................. 47
   3.5.4 Constrained Clustering Accuracy ............................... 47
   3.5.5 Clustering Efficiency ............................................. 49
3.6 Summary .......................................................................... 51
3.7 Proofs .............................................................................. 52
   3.7.1 Optimal Cluster Centroids and Weights ....................... 52
   3.7.2 Convergence of Locally Weighted Clustering ................ 54
   3.7.3 Error Probability of One Chunklet .............................. 55
   3.7.4 Error Probability of Two Chunklets ............................ 58

CHAPTER 4: SUBSPACE PROJECTION AND ITS OPTIMIZATION 61

4.1 Introduction ................................................................. 61
4.2 Related Work ............................................................... 62
4.3 Dimensionality Reduction .............................................. 64
   4.3.1 Preliminaries ....................................................... 64
   4.3.2 Feature Extraction ................................................. 65
5.5 Efficient Query Evaluation ............................................. 103
  5.5.1 The First Lower Bounding ........................................ 103
  5.5.2 The Second Lower Bounding ................................... 104
  5.5.3 Minimum Bounding Rectangle Approximation .............. 105
  5.5.4 Multi-Step Nearest Neighbor Search ......................... 110
  5.5.5 Early-Stop Mechanism ........................................ 112

5.6 Experimental Results ..................................................... 113
  5.6.1 Settings ................................................................ 113
  5.6.2 Retrieval Accuracy of Distance Metrics .................... 114
  5.6.3 Efficiency of Query Processing ............................... 116

5.7 Summary ................................................................. 118

5.8 Proofs ................................................................. 119
  5.8.1 Maximum Likelihood Approach .............................. 119
  5.8.2 Approximation Ratio of Algorithm 5.3 ..................... 120
  5.8.3 Lower Bounding of Algorithm 5.4 ......................... 121

CHAPTER 6: AUTOMATIC FEATURE GENERATION FOR MULTIPLE
INSTANCE LEARNING ......................................................... 124

6.1 Introduction .......................................................... 124

6.2 Related Works .......................................................... 127

6.3 Overview of the Proposed Algorithm ............................ 128

6.4 The Categorization Process in Details .......................... 130
  6.4.1 Outlier Removal ................................................ 130
  6.4.2 Instance-level Clustering ..................................... 132
  6.4.3 Category Pattern Mining ...................................... 133
  6.4.4 Pattern Feature Generation ................................. 136
  6.4.5 Bag-Pattern Distance Computation ......................... 137
6.4.6 Multi-Class Classification ........................................... 140
6.5 Categorization Performance ........................................... 141
  6.5.1 Experimental Settings ........................................... 141
  6.5.2 Categorization Performance ................................... 142
6.6 Summary ............................................................... 145
6.7 Proofs ................................................................. 146

CHAPTER 7: DISCUSSIONS .................................................. 148
  7.1 Dimension Reduction and Manifold Learning ..................... 148
  7.2 Similarity Search in Vector Space ................................. 150
  7.3 Data Clustering ....................................................... 152
  7.4 Design of Distance Metrics ....................................... 153
  7.5 Multiple Instance Learning ....................................... 154

REFERENCES ................................................................. 173
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Projections of PCA and LPP on the two toy datasets.</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Projections of LGSPP on the two toy datasets.</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>Projections of LGSPP on synthetic datasets I.</td>
<td>19</td>
</tr>
<tr>
<td>2.4</td>
<td>Projections of LGSPP on synthetic datasets II.</td>
<td>20</td>
</tr>
<tr>
<td>3.1</td>
<td>Accuracy of semi-supervised clustering algorithms.</td>
<td>50</td>
</tr>
<tr>
<td>3.2</td>
<td>Constraints violation of semi-supervised clustering algorithms.</td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>Projections of vector $\vec{x}$</td>
<td>66</td>
</tr>
<tr>
<td>4.2</td>
<td>Histograms on average selectivity under different query thresholds</td>
<td>71</td>
</tr>
<tr>
<td>4.3</td>
<td>Relationships between selectivity and $sv$, $sc$</td>
<td>75</td>
</tr>
<tr>
<td>4.4</td>
<td>SSP before and after GreedySC</td>
<td>82</td>
</tr>
<tr>
<td>4.5</td>
<td>Selectivity of different techniques</td>
<td>85</td>
</tr>
<tr>
<td>4.6</td>
<td>IO access counter of different techniques</td>
<td>86</td>
</tr>
<tr>
<td>4.7</td>
<td>Geometric viewpoint</td>
<td>87</td>
</tr>
<tr>
<td>5.1</td>
<td>Toy examples.</td>
<td>95</td>
</tr>
<tr>
<td>5.2</td>
<td>Feature vectors of the segmented regions in Images $I1$, $I2$ and $I3$.</td>
<td>95</td>
</tr>
<tr>
<td>5.3</td>
<td>Average retrieval accuracy of different distance metrics.</td>
<td>115</td>
</tr>
<tr>
<td>5.4</td>
<td>Efficiency of different query evaluation techniques.</td>
<td>117</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

2.1 Datasets used in the experiments ........................................... 22

2.2 Clustering quality of different embedding techniques ................. 24

3.1 Datasets used in the experiments ........................................... 46

3.2 Accuracy of unsupervised clustering algorithms. ....................... 48

3.3 Accuracy of semi-supervised clustering algorithms. ................... 49

3.4 Average number of iterations. ............................................. 52

4.1 Correlation of selectivity and $sv$, $sc$ .................................. 74

4.2 Datasets used in the experiments ........................................... 80

4.3 Correlation of selectivity, $sv$ and $sc$ .................................. 81

4.4 Performance of the greedy solution ....................................... 83

5.1 Datasets used in the experiments. ......................................... 113

5.2 Average retrieval accuracy of different distance metrics. .......... 116

6.1 Twenty image categories and the average number of instances per bag for each category ...................................................... 142

6.2 The number of patterns and the average number of instances per pattern for each category on 1000-Image ................................. 144

6.3 Average categorization accuracy of different techniques ............. 144

6.4 The confusion matrix of BP-MLP on 1000-Image ....................... 145

6.5 Average elapsed time (in seconds) to train the classification models by the proposed techniques .............................................. 145
LIST OF ACRONYMS

This is optional.
CHAPTER 1: INTRODUCTION

This chapter first lists my research interests and all my published papers. I briefly discuss the motivations of my dissertation research and present the contributions of my study.

1.1 Publications

My research interests include database, data mining, multimedia information retrieval, storage systems and Internet applications. Thirteen referred papers have been published [48,44, 49,46,51,50,47,43,95,124,125,122,142] and one paper is currently under preparation [45], which are arranged under different topics as follows:

**Dimension Reduction and Manifold Learning**


(iv) **Hao Cheng**, Kien A. Hua and Khanh Vu. “Local and Global Structures Preserving Projection.” In *Proceedings of the 19th IEEE International Conference on Tools with

Constrained Clustering


Image Retrieval


Point-Set Similarity Search and Multiple Instance Learning


Large Scale Systems


1.2 Motivations and Contributions

I am interested in different kind of algorithms and applications related to similarity search, which essentially is to retrieve objects from a dataset similar to the user selected object. Similarity query and ranking are of fundamental importance in various applications, such as time series [67,116], shapes matching [79], image, video, text retrieval [123,118] and so on. In general, there are two main aspects in this research direction: accuracy and efficiency.
The accuracy problem is how to effectively identify results which can better match the ground truth. The retrieval, clustering and classification problems are related and practical in many application domains. Objects are usually represented as high-dimensional vectors, but they may inherently reside in some low-dimensional meaningful manifolds [46]. It is interesting to extract the manifold structures of the data, which can help visualize the data, understand the underlying forces of the data, reduce storage overhead and accelerate computations with the embedded data. The clustering problem is to partition a given dataset into a set of homogeneous clusters in a systematic way [82]. Each cluster consists of objects which appear similar to each other. The grouping of objects can be important to build index structures to speed up the retrieval process, to boost the accuracy of retrieval, and to help users to easily browse a large data repository. Hierarchical clusters of objects can be considered as a rough summarization of the whole dataset [81]. Therefore, to effectively arrange a collection of objects into salient clusters is indeed a necessity and useful to many applications. Recently, there is growing interest in the new data model in which an object is denoted as a point set [96]. The corresponding retrieval and learning problems are important yet challenging.

The efficiency problem is how to process users’ queries as fast as possible. Similarity search is to retrieve objects considered ‘similar’ to the object of interest within some user-specified threshold. The Euclidean distance is the most common model to measure the (dis)similarity between objects [124] and many other metrics are designed based on it [52, 70, 105]. If the size of the dataset is larger and the dimensionality is high, to process similarity query efficiently is a challenging problem due to the so-called curse of dimensionality [113], which causes many indexing techniques to perform poorly [133]. Various dimension reduction methods have been proposed in the literature to address this problem, however, most of them are difficult to implement and still computational expensive.

In summary, the research problems related to similarity search are indeed non-trivial because of the complexity of the high-level semantic concepts, the heterogeneous natures
of the feature space, the high dimensionality of data representations, and the size of the dataset. My dissertation is dedicated to these research issues. Specifically, my work has five main contributions as below.

The first contribution is a novel manifold learning algorithm, *Local and Global Structures Preserving Projection (LGSPP)*, which defines salient low-dimensional representations for the high-dimensional data. A small number of projection directions are sought in order to properly preserve the local and global structures for the original data. Specifically, two groups of points are extracted for each individual point in the dataset: the first group contains the nearest neighbors of the point, and the other set are a few sampled points far away from the point. These two point sets respectively characterize the local and global structures with regard to the data point. The objective of the embedding is to minimize the distances of the points in each local neighborhood and also to disperse the points far away from their respective remote points in the original space. In this way, the relationships between the data in the original space are well preserved with little distortions.

The second contribution is a new constrained clustering algorithm. Conventionally, clustering is a unsupervised learning problem, which systematically partitions a dataset into a small set of clusters such that data in each cluster appear similar to each other compared with those in other clusters. In the proposal, the partial human knowledge is exploited to find better clustering results. Two kinds of constraints are integrated into the clustering algorithm. One is the must-link constraint, indicating that the involved two points belong to the same cluster. On the other hand, the cannot-link constraint denotes that two points are not within the same cluster. Given the input constraints, data points are arranged into small groups and a graph is constructed to preserve the semantic relations between these groups. The assignment procedure makes a best effort to assign each group to a feasible cluster without violating the constraints. The theoretical analysis reveals that the probability of data points being assigned to the true clusters is much higher by the new proposal, compared to conventional methods. In general, the new scheme can produce clusters which can better
match the ground truth and respect the semantic relations between points inferred from the constraints.

The third contribution is a unified framework for partition-based dimension reduction techniques, which allows efficient similarity retrieval in the high-dimensional data space. Recent similarity search techniques, such as Piecwise Aggregate Approximation (PAA), Segmented Means (SMEAN) and Mean-Standard deviation (MS), prove to be very effective in reducing data dimensionality by partitioning dimensions into subsets and extracting aggregate values from each dimension subset. These partition-based techniques have many advantages including very efficient multi-phased pruning while being simple to implement. They, however, are not adaptive to different characteristics of data in diverse applications. In this study, a unified framework for these partition-based techniques is proposed and the issue of dimension partitions is examined in this framework. An investigation of the relationships of query selectivity and the dimension partition schemes discovers indicators which can predict the performance of a partitioning setting. Accordingly, a greedy algorithm is designed to effectively determine a good partitioning of data dimensions so that the performance of the reduction technique is robust with regard to different datasets.

The fourth contribution is an effective similarity search technique in the database of point sets. In the conventional model, an object corresponds to a single vector. In the proposed study, an object is represented by a set of points. In general, this new representation can be used in many real-world applications and carries much more local information, but the retrieval and learning problems become very challenging. The Hausdorff distance is the common distance function to measure the similarity between two point sets, however, this metric is sensitive to outliers in the data. To address this issue, a novel similarity function is defined to better capture the proximity of two objects, in which an one-to-one mapping is established between vectors of the two objects. The optimal mapping minimizes the sum of distances between each paired points. The overall distance of the optimal matching is robust and has high retrieval accuracy. The computation of the new distance function is
formulated into the classical assignment problem. The lower-bounding techniques and early-stop mechanism are also proposed to significantly accelerate the expensive similarity search process.

The classification problem over the point-set data is called *Multiple Instance Learning* (MIL) in the machine learning community in which a vector is an instance and an object is a bag of instances. The fifth contribution is to convert the MIL problem into a standard supervised learning in the conventional vector space. Specially, feature vectors of bags are grouped into clusters. Each object is then denoted as a bag of cluster labels, and common patterns of each category are discovered, each of which is further reconstructed into a bag of features. Accordingly, a bag is effectively mapped into a feature space defined by the distances from this bag to all the derived patterns. The standard supervised learning algorithms can be applied to classify objects into pre-defined categories. The results demonstrate that the proposal has better classification accuracy compared to other state-of-the-art techniques.

In future, I would like to direct my research in practical applications. I am very interested in working on large-scale data analysis algorithms, applications and system developments.

### 1.3 Overview of the Dissertation

The rest of this proposal is organized as follows. Chapter 2 presents the effective manifold learning algorithm, Local and Global Structures Preserving Projection. Chapter 3 discusses the constrained clustering algorithms, guided by the input constraints. Chapter 4 presents the generic framework for a class of partition-based dimension reduction techniques. Chapter 5 introduces the effective similarity retrieval distance function for the point-set data. Chapter 6 presents a novel scheme to convert the multiple instance learning problem into the standard supervised learning problem. Chapter 7 discusses the findings in the dissertation research.
CHAPTER 2: LOCAL AND GLOBAL STRUCTURES
PRESERVING PROJECTION

Manifold learning addresses the problem to discover the intrinsic low dimensional structure from a collection of points, which reside in a high dimensional space [37]. This topic is a promising direction in dimensionality reduction and has attracted much research attention from various Computer Science communities because manifold learning can be applied to visualize the data, understand the underlying forces of the data, reduce storage overhead and accelerate computations using the embedded data [78].

Formally, let $\vec{x}$ be a vector in the $m$-dimensional data space $\mathbb{R}^m$, whose $i$th component is $x_i$. Denote its low-dimensional representation to be $P\vec{x} \in \mathbb{R}^l$, ($l \ll m$), the $i$th component of which is $P_i\vec{x}$. Manifold learning and dimensionality reduction is to derive an appropriate embedding function which maps a given data point $\vec{x}$ onto its $P\vec{x}$ [26,37].

2.1 Introduction

This section first discusses the works related to our research and then present the motivation of our proposal.

2.1.1 Related Work

There have been many proposals in dimensionality reduction and manifold learning in the literature. One of the classical methods is Principal Component Analysis (PCA) [60], targeting to find a set of axes along which the data exhibit greater variances than any other axes. The original data are projected onto the new axes in order to minimize information loss caused by the reduction. Since it optimizes the mapping based on the global correlations in the given dataset, PCA is likely to distort the local correlation structures of the data. To address this problem, Locality Preserving Projection (LPP) [75,147] encodes the local
neighborhood information into a similarity matrix and derives a linear manifold embedding of much lower dimensionality as the optimal approximation to this neighborhood structure. However, LPP may overlook the global structure of the data. Both PCA and LPP are linear methods, whose embeddings are based on their respective global linear transformation functions.

There are also many nonlinear embedding methods. In ISOMAP \([119,120]\), the points are connected with their respective neighbors to build a graph and then a pairwise distance matrix is computed based on the graph distances between the points. After that, Multidimensional Scaling (MDS) \([30]\) is applied to compute low dimensional coordinates of data points to minimize the distortion of the pairwise distances after the embedding. In Locally Linear Embedding (LLE) \([109,111]\), each data point is represented as a weighted combination of points in its neighborhood. The smooth local geometry of the data is captured in the weight matrix. LLE finds an embedding that agrees with the extracted local geometry information. Hessian LLE (HLLE) \([62]\) and Local Tangent Space Alignment (LTSA) \([144]\) find the embeddings according to the tangent space around each individual data point. The recent proposal, Diffusion Maps (DM) \([99]\) assumes data are randomly sampled from a family of random walk processes with respect to diffusion kernels, and the corresponding principal components are derived, which characterize the underlying geometrical structures of the data. Generally, nonlinear algorithms are computational intensive and do not scale very well \([75]\). They perform well on the ‘well behaved’ data, but their performances on the real-world problems are found not very robust according to several independent research groups \([37,77,139]\).

### 2.1.2 Motivation Of Our Approach

Below we would like to discuss two toy examples, which motivate our study. Figures 2.1.c - 2.1.f are the applications of PCA and LPP over the toy datasets in Figs. 2.1.a and 2.1.b.
There are 2 classes of data in each toy dataset, which are denoted by different colors. LPP finds the direction with little distortion of local structures, thus it outperforms PCA in the 1st case: the projections of the 2-class data along the principal axis of PCA are overlapped, while those of LPP are well separated. On the contrary, it is desired to preserve the global structure of the data in the 2nd case, and therefore PCA is better. Either of these two methods can only separate distinct classes of data in one of the two datasets. These examples tell us that it is important to properly consider both the local and global structures in manifold learning, which is the motivation of our algorithm, Local and Global Structures Preserving Projection (LGSPP). We highlight several aspects of the new proposal below:

(i) LGSPP finds a set of axes with little distortion of the local and global structures of the data.

(ii) LGSPP is a linear method, and therefore it is fast, robust and suitable for large scale applications.

(iii) LGSPP is defined everywhere. The projection coordinates of any new data points can be simply derived.

The remainder of this chapter is organized as follows. Section 2.2 presents our algorithm and the theoretical ideas are discussed in Section 2.3. The experimental results are reported in Section 2.4. Finally we present the summary of the chapter in Section 2.5.

### 2.2 Local and Global Structures Extraction

LGSPP targets to find a low-dimensional feature space which preserves the local structure of the data without the distortion of the global structure. Therefore, the local and global structure information has to be extracted first, and this process is discussed below.

To extract the local neighborhood information, for each data point $\vec{x}$ in the dataset, compute its $K_s$ nearest neighbors according to Euclidean distance in the original input
Figure 2.1: Projections of PCA and LPP on the two toy datasets.
space. Define $S(\vec{x})$ to be the set of points including $\vec{x}$ itself and its $K_s$ neighbors. In general, a point and its nearest neighbors are found to be ‘similar’. It is inferred that the points in each $S(\vec{x})$ are similar to each other. In this way, the local neighborhood relationship is captured in $S(\vec{x})$ of all the data points $\vec{x}$.

With regard to the global structure, for each point $\vec{x}$, we would like to select $K_d$ points from the dataset, denoted as $D(\vec{x})$, such that these $K_d$ points are far from $\vec{x}$ and also far from each other in the set. This design is inspired by the fact that the points which are not near $\vec{x}$ are generally semantically different from $\vec{x}$. At the same time, it is desirable that the chosen data points can be distinguishable to each other as much as possible. In other words, the points in $D(\vec{x})$ are expected to come from different regions of the data space and therefore they are likely to belong to distinct classes. Algorithm 2.1 is proposed to effectively sample a desirable set $D(\vec{x})$ from the dataset.

**Algorithm 2.1** Find $K_d$ remote points of $\vec{x}$ in the dataset

**Require:** the dataset $X$, the point $\vec{x}$, and the number of remote points $K_d$.

**Ensure:** the set $D(\vec{x})$ of $K_d$ remote points.

1: Randomly select $\lceil c(K_d + 1) \ln(K_d + 1) \rceil$ points from the entire dataset, which is denoted as $S$.

2: while there are more remote points to be found do

3: Find the point $\vec{f}$ in $S$, which has the largest minimum distance to all previously selected points in $D(\vec{x})$ and also $\vec{x}$, i.e.,

$$\arg \max_{\vec{f} \in S} \left( \min_{\vec{y} \in \{\vec{x}, D(\vec{x})\}} \sum_{i=1}^{m} (f_i - y_i)^2 \right).$$

4: Insert $\vec{f}$ in $D(\vec{x})$.

5: end while

The first step of the algorithm is to sample a subset of points from the entire dataset. The sampling procedure is necessary to avoid the selection of outliers because:

(i) In general, the outliers are not the representatives of data and carry little useful information in preserving the global structures.
(ii) The outliers are usually far from any other points in the dataset and tend to be selected by Algorithm 2.1 if the sampling step is not used.

The size of the samples is \( [c(K_d + 1) \ln(K_d + 1)] \), where the constant \( c \) is set to be 3, which works well in this study. The sample size is determined in this way such that the samples can come from more than \( K_d \) different remote neighborhoods with high probability (the proof is available in Appendix in [121]). After getting the sample set, we first choose the point which is farthest from \( \vec{x} \), and in the consecutive steps, the points which are far away from the already selected ones and \( \vec{x} \), are chosen and inserted into \( D(\vec{x}) \).

After computing the sets \( S(\vec{x}) \) and \( D(\vec{x}) \) for each \( \vec{x} \), the last step is compute the embedding. For each point \( \vec{x} \), \( S(\vec{x}) \) contains the points in the neighborhood of \( \vec{x} \), and the points in \( D(\vec{x}) \) are generally from multiple distinct neighborhoods and also different from those in \( S(\vec{x}) \). Therefore, the embedding process should find a reduced space such that the points in \( S(\vec{x}) \) are kept close and the two point sets \( S(\vec{x}) \) and \( D(\vec{x}) \) are mapped separated. In summary, the whole procedure of LGSPS is shown as follows:

(i) **Collect local neighborhood structure information:** For each \( \vec{x} \), compute \( S(\vec{x}) \) including \( \vec{x} \) itself and its \( K_s \) nearest neighbors.

(ii) **Collect global structure information:** For each \( \vec{x} \), use Algorithm 2.1 to compute the set of \( K_d \) remote points of \( \vec{x} \), which is \( D(\vec{x}) \).

(iii) **Embedding:** Derive a low-dimensional embedding in order to minimize the sum of pairwise distances of the points in each \( S(\vec{x}) \), and in the mean time maximize the distances between the points in \( S(\vec{x}) \) and \( D(\vec{x}) \).

In the next subsection, we further discuss the underlying theoretical meaning of the above design and how we can formulate and solve the above embedding problem (Step 3) effectively.
2.3 Optimal Projection

2.3.1 Problem Formulation

LGSPP is a linear algorithm, whose mapping depends on a set of $l$ axes $\vec{p}_i$. For a given data point $\vec{x}$, its embedding coordinates are the projections of $\vec{x}$ onto the axes, which are $P_i \vec{x} = \sum_{j=1}^{m} x_j p_{ij}$, for $1 \leq i \leq l$.

The purpose of the embedding is to find the salient representations of the data. Because the points in $S(\vec{x})$ are considered mutually similar, these points should be mapped onto a small local neighborhood in the feature space: their distances are expected to be as small as possible. Therefore, the first objective is to minimize the sum of pairwise squared distances in $S(\vec{x})$:

$$\min \sum_{\vec{x} \in X} \sum_{\vec{y} \in S(\vec{x})} \sum_{\vec{z} \in S(\vec{x})} \sum_{i=1}^{m} (P_i \vec{y} - P_i \vec{z})^2.$$

As the points in $D(\vec{x})$ carry the information of multiple neighborhoods except the one near $S(\vec{x})$, it is desired to have the embedding to maintain this global structure, i.e., the neighborhood of $S(\vec{x})$ should be far from those of $D(\vec{x})$. Therefore the second objective is to maximize the sum of distances between the points in $S(\vec{x})$ and $D(\vec{x})$:

$$\max \sum_{\vec{x} \in X} \sum_{\vec{y} \in S(\vec{x})} \sum_{\vec{z} \in D(\vec{x})} \sum_{i=1}^{m} (P_i \vec{y} - P_i \vec{z})^2.$$

For LGSPP, the optimal set of axes should achieve the above defined two objectives in order to preserve the local and global structures of the data. We design to find the axis $\vec{p}_i$ one by one which optimizes the unified objective, i.e., minimizes the ratio of the 1st minimum objective to the 2nd maximum one:

$$\arg \min_{\vec{p}_i} \frac{\sum_{\vec{x} \in X} \sum_{\vec{y} \in S(\vec{x})} \sum_{\vec{z} \in S(\vec{x})} (P_i \vec{y} - P_i \vec{z})^2}{\sum_{\vec{x} \in X} \sum_{\vec{y} \in S(\vec{x})} \sum_{\vec{z} \in D(\vec{x})} (P_i \vec{y} - P_i \vec{z})^2}.$$
As the magnitude of $\vec{p}_i$ does not have impact on the objective, the above problem is essentially equivalent to

$$\arg\min_{\vec{p}_i} \sum_{\vec{x} \in X} \sum_{\vec{y} \in S(\vec{x})} \sum_{\vec{z} \in S(\vec{y})} (P_{\vec{y}}^i - P_{\vec{z}}^i)^2,$$

(Eq. 2.1)

subject to

$$\sum_{\vec{x} \in X} \sum_{\vec{y} \in S(\vec{x})} \sum_{\vec{z} \in S(\vec{y})} (P_{\vec{y}}^i - P_{\vec{z}}^i)^2 = 1.$$

(Eq. 2.2)

Intuitively, Eq. 2.1 reflects our goal to preserve the local neighborhood structures. The constraint in Eq. 2.2 can avoid the trivial set of axes: the vector of zero magnitude would cause the entire input space to collapse into a single point. More importantly, the constraint actually aims to preserve the global structure of the data: the local neighborhood ($S(\vec{x})$) and the remote neighborhoods ($D(\vec{x})$) are prevented from being close in the reduced space. Note that Eq. 2.1 and Eq. 2.2 are implicit functions of the parameter $\vec{p}_i$. In next subsection, we will rewrite them into the explicit forms of $\vec{p}_i$ and provide a solution to the formulated problem.

### 2.3.2 Problem Solution

We first introduce the notations, which will be used in the following rewriting. For a given subset of points $S$, the mean of is an m-dimensional column vector, denoted as $M(S)$, whose $i$th component is

$$M_i(S) = \frac{1}{|S|} \sum_{\vec{x} \in S} x_i,$$
and its covariance matrix $C(S)$ is an $m \times m$ matrix, in which, for $1 \leq i, j \leq m$,

$$C_{ij}(S) = \frac{1}{|S|} \left( \sum_{\vec{x} \in S} x_i x_j - \frac{1}{|S|} \left( \sum_{\vec{x} \in S} x_i \right) \frac{1}{|S|} \left( \sum_{\vec{x} \in S} x_j \right) \right) = \frac{1}{|S|} \sum_{\vec{x} \in S} x_i x_j - M_i(S) M_j(S).$$

For two subsets $S_1$ and $S_2$, define an $m \times m$ matrix $M(S_1, S_2)$, where

$$M(S_1, S_2) = (M(S_1) - M(S_2))(M(S_1) - M(S_2))^T.$$

$T$ is the transpose of vector. Accordingly, we can rewrite the objective function as follows (the proofs are available in Section 2.6 at the end of this chapter):

$$\sum_{\vec{x} \in X} \sum_{\vec{y} \in S(\vec{x})} \sum_{\vec{z} \in S(\vec{x})} (P_{\vec{y}}^i - P_{\vec{z}}^i)^2, = 2\vec{p}_i^T \left( \sum_{\vec{x} \in X} |S(\vec{x})| C(S(\vec{x})) \right) \vec{p}_i,$n

and similarly, the constraint is equivalent to

$$\sum_{\vec{x} \in X} \sum_{\vec{y} \in S(\vec{x})} \sum_{\vec{z} \in D(\vec{x})} (P_{\vec{y}}^i - P_{\vec{z}}^i)^2, = \vec{p}_i^T \left( \sum_{\vec{x} \in X} |D(\vec{x})| (C(S(\vec{x})) + C(D(\vec{x})) + M(S(\vec{x}), D(\vec{x}))) \right) \vec{p}_i.$n

Now both the goal and the constraint are the explicit functions of the axis $\vec{p}_i$. Hence the problem to be solve is to find the axis such that

$$\arg \min_{\vec{p}_i} \vec{p}_i^T A \vec{p}_i$$

subject to

$$\vec{p}_i^T B \vec{p}_i = 1.$$
where

\[
A = 2 \sum_{\vec{x} \in X} |S(\vec{x})|^2 C(S(\vec{x})) \\
B = \sum_{\vec{x} \in X} |S(\vec{x})||D(\vec{x})|(C(S(\vec{x}))+C(D(\vec{x}))+M(S(\vec{x}), D(\vec{x})))
\]

The above problem can be solved using the \textit{Lagrange Multipliers method}:

\[
\frac{\partial}{\partial \vec{p}_i} \vec{p}_i^T A \vec{p}_i - \lambda_i (\vec{p}_i^T B \vec{p}_i - 1) = A \vec{p}_i - \lambda_i B \vec{p}_i = 0.
\]

It is easy to show that the matrices $A$ and $B$ are both symmetric and positive-semidefinite. Therefore a set of axes $\vec{p}_1, \ldots \vec{p}_l$ are corresponding to the eigenvectors of the top $l$ smallest eigenvalues for the generalized eigenvalue problem $A \vec{p}_i = \lambda_i B \vec{p}_i$, which can be solved effectively via the conventional mathematical computation tool [14].

Let us review the whole procedure to compute LGSPP embedding again. For each point $\vec{x}$, its neighborhood set $S(\vec{x})$ and remote set $D(\vec{x})$ are first determined. Sum up all the covariance matrices $C(S(\vec{x}))$ of all $\vec{x}$ to get the matrix $A$. The matrix $B$ is computed based on the summations of $C(S(\vec{x}))$, $C(D(\vec{x}))$ and $M(S(\vec{x}), D(\vec{x}))$ for all $\vec{x}$. After that, to find an optimal embedding is essentially a generalized eigenvalue problem.

### 2.4 Experimental Results

In this subsection, we first show the performance of our LGSPP on synthetic datasets and then compare it against the well-known methods over the real-world datasets.

#### 2.4.1 Synthetic Data Examples

Figures 2.2.a and 2.2.b respectively plot the projection data along the first dimension determined by LGSPP for the two toy datasets in Section 2.1.2. Recall that neither PCA nor LPP
does well on both examples. It is interesting to observe that our method performs well in both datasets: two classes of data are mapped far apart (without any overlap). In the principal direction determined by LGSPP, different classes of data can be easily distinguished.

![Figures 2.2: Projections of LGSPP on the two toy datasets.](image)

We applied LGSPP over the synthetic datasets used in [144]. The color-coded data in Figure 2.3.a are 400 points equally sampled from the data generating function:

\[ g(t) = (10t, 10t^3 + 2t^2 - 10t) , t \in [-1, 1] \]

in which \( t \) is the free variable in the range from -1 to 1. The corresponding figure on the right, Fig. 2.3.b plots the data of the 1st reduced dimension of LGSPP versus the governing force of the original data, which is \( t \). Ideally, the reduced data and \( t \) should form a perfect straight line. It is observed from the plot that the LGSPP can indeed discover the underlying force of the data. Similarly, the data in Figs. 2.3.c, 2.4.a and 2.4.c are respectively sampled
from the following functions:

\[
g(t) = (10 \cos(t), \sin(t)), \quad t \in \left[\frac{\pi}{2}, \frac{3\pi}{2}\right]
\]

\[
g(t) = (\cos(t)^3, \sin(t)^3), \quad t \in [0, \pi]
\]

\[
g(t) = (3 \cos(t), 3 \sin(t), 3t), \quad t \in [0, 4\pi]
\]

and Fig. 2.4.e plots the Gaussian random samples. The corresponding reduced coordinates are plotted against \(t\) on the figures of the right hand side, all of which are approximately
Figure 2.4: Projections of LGSPP on synthetic datasets II.
straight lines. This shows that LGSPP can find the right principal direction of the given data, which represents the governing force of the data.

2.4.2 Clustering Evaluation

We compared our LGSPP against other embedding methods by clustering [81]. K-Means clustering [73] was executed on the reduced space generated respectively by PCA [60], LPP [147], Diffusion Maps (DM) [99] and LGSPP. Besides DM, we did try other nonlinear methods such as LLE, ISOMAP, HLLE and so on. We chose DM as the representative of the nonlinear algorithms because of its relatively good and reliable performances over the test datasets. We also compared the above methods with the baseline, K-Means over the original space.

In the experiments, the number of nearest neighbors was set to be 5 for all datasets. LPP used this parameter to construct the similarity matrix and in LGSPP this number is $K_s$. As the optimal number of clusters $K$ is known for each dataset, we used them in our experiments. For LGSP, we set $K_d$ to be $K - 1$. Extensive experiments were carried out over the datasets in Table 2.1. Most datasets were directly got from the UCI repository [100], where the Digits and Letter datasets were sampled by respectively extracting characters 3, 8, 9 and A, B. The Protein dataset were used in [137].

We used two common metrics to evaluate the qualities of clustering outputs of different methods. The first metric is the Rand Index [107]. Let $\phi_g$ denote the membership function of data points in the dataset according to the ground truth. In other words, $\phi_g(\vec{x})$ represents the true cluster label for $\vec{x}$. Define the binary relation $R_g$ for any pair of data points to be either 1 if they both belong to the same cluster or 0 otherwise:

\[
R_g(\vec{x}_i, \vec{x}_j) = \begin{cases} 
1, & \text{if } \phi_g(\vec{x}_i) = \phi_g(\vec{x}_j), \\
0, & \text{otherwise.}
\end{cases}
\]  

(Eq. 2.3)
<table>
<thead>
<tr>
<th>dataset</th>
<th>N</th>
<th>m</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soybean Small</td>
<td>47</td>
<td>35</td>
<td>4</td>
</tr>
<tr>
<td>Protein</td>
<td>116</td>
<td>20</td>
<td>6</td>
</tr>
<tr>
<td>Iris Plant</td>
<td>150</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Wine Recognition</td>
<td>178</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>Sonar</td>
<td>208</td>
<td>60</td>
<td>2</td>
</tr>
<tr>
<td>Heart Stat Log</td>
<td>270</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>351</td>
<td>34</td>
<td>2</td>
</tr>
<tr>
<td>Balance Scale</td>
<td>625</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Breast</td>
<td>683</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Wisconsin Breast Cancer</td>
<td>683</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Vehicle</td>
<td>846</td>
<td>18</td>
<td>4</td>
</tr>
<tr>
<td>Digits (3,8,9)</td>
<td>1008</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>Letter (A,B)</td>
<td>1555</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>Satellite Images</td>
<td>6435</td>
<td>37</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 2.1: Datasets used in the experiments

For a dataset of $N$ points, there are $\frac{(N-1)\times N}{2}$ unique pairs of relations in $\mathbb{R}_g$ between different points. Given the membership $\phi_c(\bar{x})$ for each point $\bar{x}$ by a clustering algorithm, a pairwise relation $\mathbb{R}_c$ is defined for each pair of points, similar to Eq. 2.3. Then the Rand index is the percentage of pairs in the relations $\mathbb{R}_g$ and $\mathbb{R}_c$, which agree with each other, i.e.,

$$\text{Rand}(\phi_g, \phi_c) = \frac{\sum_{i=1}^{N} \sum_{j=(i+1)}^{N} \mathbf{1}(\mathbb{R}_g(\bar{x}_i, \bar{x}_j) - \mathbb{R}_c(\bar{x}_i, \bar{x}_j))}{\binom{N}{2}},$$

(Eq. 2.4)

in which $\mathbf{1}(x)$ is the indicator function, equal to 1 if $x = 0$, and 0 otherwise. The second metric is the Normalized Mutual Information [19,147], which measures the consistency of the clustering output compared to the ground truth. It reaches the maximum value of 1 only if $\phi_c$ perfectly matches $\phi_g$ and the minimal zero if the assignments of $\phi_c$ and $\phi_g$ are independent. Formally,

$$\text{NMI}(\phi_g, \phi_c) = \frac{\sum_{i=1}^{K} \sum_{j=1}^{K} \log \frac{p_{g,c}(i,j)}{p_g(i) p_c(j)}}{\min(\sum_{i=1}^{K} p_g(i) \log \frac{1}{p_g(i)}, \sum_{j=1}^{K} p_c(j) \log \frac{1}{p_c(j)})},$$

(Eq. 2.5)
where \( p_g(i) \) is the percentage of points in Cluster \( i \) according to the ground truth, i.e. \( p_g(i) = \frac{\sum_{k=1}^{N} 1(\phi_g(\vec{x}_k)-i)}{N} \). Similarly, \( p_c(j) = \frac{\sum_{k=1}^{N} 1(\phi_c(\vec{x}_k)-j)}{N} \) and \( p_{g,c}(i, j) \) is the percentage of points that belong to Cluster \( i \) in \( \phi_g \) and also Cluster \( j \) in \( \phi_c \), i.e. \( p_{g,c}(i, j) = \frac{\sum_{k=1}^{N} 1(\phi_g(\vec{x}_k)-i)1(\phi_c(\vec{x}_k)-j)}{N} \).

We got the implementations of LPP and DM respectively from [12, 136]. For the DM, we used the default parameter: Laplace-Beltrami operator with sigma value equal to 10. For all the embedding methods, we tested the clustering quality with different reduced dimensionalities. In each test, K-Means was run 100 times with different initializations and the average performance was recorded.

For each method, we summarize their respective best performances (under the dimensionality with the largest average Rand index) in Table 2.2. As in the first row, for the Soybean dataset, the Rand index (or NMI) is 0.847 (or 0.788) as the baseline performance, K-Means over the full dimensional space; the Rand index (or NMI) is 0.851 (or 0.798) for the clustering in the 9-dimensional space defined by PCA; and LGSPP outperformed others when the embedding dimensionality is only 3. Among all the methods, the one performed the best on different datasets with regarding to a particular evaluation metric is highlighted (boldface).

In general, the two evaluation metrics are fairly consistent. It is interesting to see that different reduction methods generated much lower dimensional spaces for the datasets while having competitive or even better clustering performance compared to the baseline method. Although no single method can outperform all the others for all the datasets, the proposed LGSPP is effective in many cases. According to Rand index (or NMI), LGSSP has the best clustering performances on 8 (7 for NMI) datasets. For the other datasets, it usually performed fairly close to the best method. Therefore the embedding space defined by LGSPP is generally good in term of the discrimination capability of different clusters. It keeps the data of the same class close in the reduced space, and the data of different classes are mapped far away from each other. In this way, the local and global structures of the data are properly preserved.
Table 2.2: Clustering quality of different embedding techniques

<table>
<thead>
<tr>
<th>Dataset</th>
<th>K-Means</th>
<th>PCA</th>
<th>LPP</th>
<th>DM</th>
<th>LGSPP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rand NMI</td>
<td>Dim</td>
<td>Rand NMI</td>
<td>Dim</td>
<td>Rand NMI</td>
</tr>
<tr>
<td>Soybean</td>
<td>0.847 0.788</td>
<td>9 0.851 0.798</td>
<td>1 0.856 0.833</td>
<td>2 0.833 0.763</td>
<td>3 0.895 0.889</td>
</tr>
<tr>
<td>Protein</td>
<td>0.774 0.392</td>
<td>3 0.787 0.400</td>
<td>5 0.765 0.335</td>
<td>1 0.742 0.332</td>
<td>15 0.778 0.421</td>
</tr>
<tr>
<td>Iris</td>
<td>0.853 0.648</td>
<td>1 0.898 0.800</td>
<td>1 0.924 0.832</td>
<td>1 0.910 0.802</td>
<td>1 0.925 0.837</td>
</tr>
<tr>
<td>Wine</td>
<td>0.709 0.436</td>
<td>3 0.710 0.436</td>
<td>1 0.710 0.440</td>
<td>6 0.628 0.311</td>
<td>1 0.710 0.438</td>
</tr>
<tr>
<td>Sonar</td>
<td>0.516 0.019</td>
<td>10 0.516 0.019</td>
<td>3 0.502 0.011</td>
<td>4 0.519 0.029</td>
<td>2 0.515 0.020</td>
</tr>
<tr>
<td>Heart</td>
<td>0.516 0.019</td>
<td>3 0.516 0.019</td>
<td>1 0.524 0.031</td>
<td>13 0.512 0.026</td>
<td>3 0.535 0.045</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>0.589 0.137</td>
<td>9 0.590 0.137</td>
<td>3 0.574 0.107</td>
<td>1 0.588 0.133</td>
<td>1 0.559 0.088</td>
</tr>
<tr>
<td>Balance</td>
<td>0.582 0.114</td>
<td>3 0.586 0.121</td>
<td>2 0.586 0.124</td>
<td>3 0.587 0.125</td>
<td>3 0.592 0.135</td>
</tr>
<tr>
<td>Breast</td>
<td>0.925 0.752</td>
<td>1 0.926 0.755</td>
<td>1 0.924 0.753</td>
<td>8 0.954 0.840</td>
<td>1 0.932 0.772</td>
</tr>
<tr>
<td>Pima cancer</td>
<td>0.551 0.033</td>
<td>1 0.552 0.033</td>
<td>3 0.552 0.037</td>
<td>1 0.546 0.138</td>
<td>2 0.563 0.052</td>
</tr>
<tr>
<td>Vehicle</td>
<td>0.645 0.194</td>
<td>3 0.647 0.193</td>
<td>2 0.644 0.182</td>
<td>1 0.643 0.204</td>
<td>3 0.649 0.192</td>
</tr>
<tr>
<td>Digits(3,8,9)</td>
<td>0.842 0.777</td>
<td>5 0.861 0.786</td>
<td>7 0.851 0.800</td>
<td>3 0.433 0.212</td>
<td>4 0.846 0.818</td>
</tr>
<tr>
<td>Letter(A,B)</td>
<td>0.777 0.474</td>
<td>5 0.777 0.474</td>
<td>1 0.890 0.731</td>
<td>3 0.838 0.629</td>
<td>4 0.896 0.746</td>
</tr>
<tr>
<td>Sat Images</td>
<td>0.836 0.574</td>
<td>13 0.843 0.590</td>
<td>5 0.845 0.593</td>
<td>5 0.605 0.480</td>
<td>8 0.843 0.583</td>
</tr>
</tbody>
</table>

2.5 Summary

In this chapter, we proposed a new manifold learning method, LGSPP, which targets to find a low-dimensional space for a given dataset with little distortion of the local and global correlation structures. For each point, LGSPP extracts the corresponding local neighborhood and a set of points sampled from the remote neighborhoods. In the reduced space, the local neighborhoods are kept as compact as possible, while the data points in the remote neighborhoods are dispersed far away. We tested LGSPP over many datasets and the results show that LGSPP can find a salient embedding space for the multi-dimensional data, in which different classes of data can be well separated.

Currently LGSPP performs fairly well compared with nonlinear methods. It may imply that most real datasets are not very complex and using the linear projection to approximate the underlying manifolds may be reasonably good. We would like to pursue further on this problem and it is also interesting to develop a nonlinear counterpart of LGSPP in the near future.
2.6 Proofs

Given two subsets \( S_1 \) and \( S_2 \), all the data points are projected onto the new set of axes \( \vec{p}_1, \vec{p}_2, \ldots, \vec{p}_l \). The sum of squared distances between all the points in \( S_1 \) and those in \( S_2 \) can be rewritten as follows:

\[
\sum_{\vec{x} \in S_1} \sum_{\vec{y} \in S_2} (P^x - P^y)^2 = |S_1||S_2| \left( \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} (P^x)^2 + \frac{1}{|S_2|} \sum_{\vec{y} \in S_2} (P^y)^2 - 2 \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} \frac{1}{|S_2|} \sum_{\vec{y} \in S_2} P^x P^y \right)
\]

\[
= |S_1||S_2| \left( \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} (P^x)^2 - \left( \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} P^x \right)^2 + \frac{1}{|S_2|} \sum_{\vec{y} \in S_2} (P^y)^2 - \left( \frac{1}{|S_2|} \sum_{\vec{y} \in S_2} P^y \right)^2 \right)
\]

\[
+ \left( \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} P^x \right)^2 - 2 \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} \frac{1}{|S_2|} \sum_{\vec{y} \in S_2} P^x \sum_{\vec{y} \in S_2} P^y + \left( \frac{1}{|S_2|} \sum_{\vec{y} \in S_2} P^y \right)^2
\]

The equations in the bracket can be split into 3 parts. As

\[ P^x = \sum_{j=1}^m p_{ij} x_j, \]

we have,

\[
\frac{1}{|S_1|} \sum_{\vec{x} \in S_1} (P^x)^2 - \left( \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} P^x \right)^2
\]

\[
= \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} \left( \sum_{j=1}^m p_{ij} x_j \right)^2 - \left( \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} \sum_{j=1}^m p_{ij} x_j \right)^2
\]

\[
= \sum_{j=1}^m \sum_{k=1}^m p_{ij} p_{ik} \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} x_j x_k - \left( \sum_{j=1}^m p_{ij} \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} x_j \right) \left( \sum_{k=1}^m p_{ik} \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} x_k \right)
\]

\[
= \sum_{j=1}^m \sum_{k=1}^m p_{ij} p_{ik} \left( \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} x_j x_k - \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} x_j \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} x_k \right)
\]

\[
= \sum_{j=1}^m \sum_{k=1}^m p_{ij} p_{ik} C_{jk}(S_1)
\]

\[
= \vec{p}_i^T C(S_1) \vec{p}_i.
\]
Similarly, we have,

\[
\frac{1}{|S_2|} \sum_{\vec{y} \in S_2} (P_i^\vec{y})^2 - \left( \frac{1}{|S_2|} \sum_{\vec{y} \in S_2} P_i^\vec{y} \right)^2 = \vec{p}_i^T C(S_2) \vec{p}_i.
\]

The third part can be rewritten as below:

\[
\left( \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} P_i^\vec{x} \right)^2 - 2 \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} P_i^\vec{x} \frac{1}{|S_2|} \sum_{\vec{y} \in S_2} P_i^\vec{y} + \left( \frac{1}{|S_2|} \sum_{\vec{y} \in S_2} P_i^\vec{y} \right)^2
\]

\[
= \left( \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} \sum_{j=1}^m p_{ij} x_j - \frac{1}{|S_2|} \sum_{\vec{y} \in S_2} \sum_{j=1}^m p_{ij} y_j \right)^2
\]

\[
= \sum_{j=1}^m \sum_{k=1}^m p_{ij} p_{ik} \left( \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} x_j - \frac{1}{|S_2|} \sum_{\vec{y} \in S_2} y_j \right) \left( \frac{1}{|S_1|} \sum_{\vec{x} \in S_1} x_k - \frac{1}{|S_2|} \sum_{\vec{y} \in S_2} y_k \right)
\]

\[
= \sum_{j=1}^m \sum_{k=1}^m p_{ij} p_{ik} M_{ij}(S_1, S_2)
\]

\[
= \vec{p}_i^T M(S_1, S_2) \vec{p}_i.
\]

Combine the above results, we have

\[
\sum_{\vec{x} \in S_1} \sum_{\vec{y} \in S_2} (P_i^\vec{x} - P_i^\vec{y})^2 = \vec{p}_i^T |S_1||S_2|(C(S_1) + C(S_2) + M(S_1, S_2)) \vec{p}_i.
\]

Therefore the sum of squared distances between two sets after the embedding is a function of the axes. We can apply the above result to rewrite the sum of squared distance in the set \( S \). If \( S_1 = S_2 = S \), then we have \( C(S_1) = C(S_2) = C(S) \) and \( M(S_1, S_2) = M(S, S) = 0 \),
therefore we have,

$$\sum_{x \in S} \sum_{\tilde{y} \in S} (P_i^x - P_i^{\tilde{y}})^2 = \tilde{p}_i^T |S|^2 (C(S) + C(S) + M(S,S)) \tilde{p}_i = 2\tilde{p}_i^T |S|^2 C(S) \tilde{p}_i.$$
CHAPTER 3: SUBSPACE CLUSTERING AND SEMI-SUPERVISED CLUSTERING

Data clustering is a difficult problem due to the complex and heterogeneous natures of multidimensional data. To improve clustering accuracy, we propose a scheme to capture the local correlation structures: associate each cluster with an independent weighting vector and embed it in the subspace spanned by an adaptive combination of the dimensions. Our clustering algorithm also takes advantage of the known pairwise instance-level constraints. The data points in the constraint set are divided into groups through inference; and each group is assigned to the feasible cluster which minimizes the sum of squared distances between all the points in the group and the corresponding centroid. Our theoretical analysis shows that the probability of points being assigned to the correct clusters is much higher by the new algorithm, compared to the conventional methods. This is confirmed by our experimental results, indicating that our design indeed produces clusters which are closer to the ground truth than clusters created by the current state-of-the-art algorithms.

3.1 Introduction

A cluster is a set of data points which share similar characteristics to one another compared to those not belonging to the cluster [82]. While the definition is fairly intuitive, it is non trivial at all to partition a multi-dimensional dataset into meaningful clusters. Such a problem has attracted much research attention from various Computer Science disciplines because clustering has many interesting and important applications [81].

In general, data objects are represented as feature vectors in clustering algorithms. Although the feature space is usually complex, it is believed that the intrinsic dimensionality of the data is generally much smaller than the original one [124]. Furthermore, the data are often heterogeneous. That is, different subsets of the data may exhibit different correlations;
and in each subset, the correlations may vary along different dimensions [104]. As a result, each feature dimension may not necessarily be uniformly important for different regions of the entire data space. These observations motivate a lot of interest in constructing a new ‘meaningful’ feature space over a given set of data. Many global dimension reduction techniques such as [60] work on the derivation of new axes in the reduced space, onto which the original data space is projected. Recent studies in manifold learning [147] embed the space onto low-dimensional manifolds in order to discover the intrinsic structure of the entire space, which have shown encouraging results. To directly tackle the heterogeneous issue, adaptive distance metrics have been proposed [61], which define the degree of similarity between data points with regard to their surrounding subspaces. Basically, the focus of the above research is to work out a new salient representation of the data in order to improve the clustering performance.

Although clustering is traditionally an unsupervised learning problem, a recent research trend is to utilize partial information to aid in the unsupervised clustering process. It has been pointed out that the pairwise instance-level constraints are accessible in many clustering practices [127], each of which indicates whether a pair of data points must reside in the same cluster or not. The constraint set is useful in two ways. One way is to learn an appropriate distance metric. The other way is to direct the algorithm to find a more suitable data partitioning by enforcing the constraints and penalizing any violations of them.

In this chapter, we propose to improve the accuracy of the clustering process in two aspects:

(i) We capture the local structures and associate each cluster with its own local weighting vector. For each cluster, a dimension along which the data values of the cluster exhibit strong correlations receives a large weight; while a small one is assigned to a dimension of large value variations.
We integrate the constrained learning into the local weighting scheme. The data points in the constraint set are arranged into disjoint groups, each assigned as a whole to a cluster according to our defined criteria.

Our experimental results as well as the theoretical analysis reveal advantages of the proposed technique.

The remainder of the chapter is organized as follows. Section 3.2 provides a survey of the related works. The locally weighted cluster concept, and the constrained learning are discussed in Section 3.3 and 3.4, respectively. The experimental results are reported in Section 3.5. Finally we conclude the chapter in Section 3.6.

### 3.2 Related Work

In this section, we will discuss the related research works in different areas, including clustering, dimension reduction, manifold learning, and constrained clustering.

There are different types of clustering algorithms, such as *partitional clustering* and *hierarchical clustering*. An example of partitional clustering is *K-Means* [73,138], in which a cluster is represented by its centroid. K-Means takes the iterative approach to minimize the sum of distances between data points and their respective nearest centroid. In hierarchical clustering, an agglomerative tree structure on a given dataset is generally created in either a *bottom-up* or *top-down* fashion. In the bottom-up approach, each data point is initially treated as a cluster by itself; and these clusters are merged in subsequent steps according to some specific criteria, such as *Single-Link*, *Complete-Link* or *Ward’s method* [84]. A limitation of these methods is that they are sensitive to outliers [145]. A representative of top-down clustering is *Bisection K-Means* [146], which starts with the entire dataset as one big cluster and iteratively picks a cluster and divides it into two parts using K-Means until the desired number of clusters has been reached. Since the clusters produced by this repeated bisection procedure tend to have relatively uniform sizes, this approach generally has a more
robust performance compared to the bottom-up clustering algorithms [145]. Recently there are also some proposals on graph theoretic clustering techniques [101,140]. Generally, they are very computationally intensive [147].

Dimension reduction techniques aim to reduce the dimensionality of the original data space. One well-known technique is Principal Component Analysis [60], which minimizes the information loss caused by the reduction. Since it optimizes the mapping based on the global correlations in the dataset, PCA is likely to distort the local correlation structures of individual clusters that might reside in different subspaces. To address this problem, the Locality Preserving Projection [147] encodes the local neighborhood information into a similarity matrix and derives a low-dimensional linear manifold embedding as the optimal approximation to this neighborhood structure. Nonetheless, this type of global transformation schemes lacks the flexibility to directly model different shapes of individual clusters. As each cluster generally is compactly embedded in a different subspace, ProClus and its generalization [16,17] seek to directly determine the subspaces for individual clusters. One disadvantage of these methods is that it may not be easy to determine the optimal dimensionality of the reduced space or the subspaces [104]. To overcome these problems, all the feature dimensions are properly weighted in the Locally Adaptive Clustering technique [61]. Specifically, the local feature selection is adopted so that different weighted distance metrics are in effect around the neighborhoods of different clusters. LAC and our local weighting scheme share the same motivation and both formulate the clustering problem as an optimization problem. However, as detailed in Section 3.3, our proposal differs in defining the objective function and the constraints. Moreover, our method does not require any tuning to control the weighting scheme and thus the performance is more stable, while that of LAC is fairly sensitive to its own tunable factor [19].

In constrained clustering, instance-level constraints indicate whether the corresponding pairs of data points belong to the same cluster or not. The constraints are usually used in learning a suitable Mahalanobis distance metric [22,137] so that the data points marked
similar are kept close to each other and the points which are identified dissimilar are dispersed far apart. The constraints are also used to directly guide the cluster assignment process. For a given set of constraints, it is desirable that a clustering algorithm does not violate any of them when producing data partitions. Constrained K-Means [128] adopted this idea and strictly enforces all the constraints over the cluster assignments. However, it has been shown that constrained clustering is a hard problem [55] and it is not necessarily a good idea to derive the partitions strictly satisfying every constraint [126]. Instead of enforcing the constraints directly, recent techniques introduced penalties on constraint violations; for example, the proposal in [55] seeks to minimize the constrained vector quantization error. The unified method, MPCK-Means [28] performs metric learning in every clustering iteration and penalizes the violations of the constraints. This technique also uses seeding to infer the initial centroids from the given constraint set to further improve the clustering performance [23]. In [72], a systematic approach is developed to tune the weights of dimensions to achieve a better clustering quality, which is defined as a weighted combination of the proportion of constraints satisfied in the output and an objective cluster validity index. Other interesting related research include the study of the utility of the constraint set [56, 57], and the modification of the Complete-Link clustering algorithm by exploring the spatial implications from the instance-level constraints [89].

In this study, we integrate the local distance metric learning with constrained learning: the locally weighting scheme can well discover clusters residing in different subspaces, and our chunklet assignment strategy aggressively utilizes the input constraints to guide the clustering process. The improvement of the clustering accuracy has been observed in our experimental study.
3.3 Locally Weighted Clustering

Let $\mathbb{R}^m$ be the $m$-dimensional data space containing a set of $N$ data points $\vec{x}_i$, whose $j$th component is $x_{ij}$. In the K-Means clustering, a cluster is represented by its centroid $\vec{c}_k \in \mathbb{R}^m$, and a given point is assigned to the closest centroid based on the Euclidean distance or some global Mahalanobis distance. As discussed before, global distance metrics are ineffective to capture the local structures.

Instead, our scheme allows different weighted distance metrics for different clusters. Specifically, besides the centroid $\vec{c}_k$, a cluster is now associated with an adaptive weighting vector $\vec{w}_k$, which is determined based on the points in this cluster. The weights $\vec{w}_k$ are used to re-scale the distance from a data point $\vec{x}$ to the centroid $\vec{c}_k$, i.e.,

$$L_{2,\vec{w}_k}(\vec{x}, \vec{c}_k) = \sqrt{\sum_{j=1}^{m} w_{kj} (c_{kj} - x_j)^2}.$$  

Each data point is placed in its nearest cluster according to the adaptive distance metric. Formally, the membership function $\phi_c$, the mapping of a point $\vec{x}$ to one of the $K$ clusters, is

$$\phi_c(\vec{x}) = \arg \min_{1 \leq k \leq K} L_{2,\vec{w}_k}(\vec{x}, \vec{c}_k). \quad (Eq. 3.1)$$

Accordingly, all the points which belong to the $k$th cluster are denoted as,

$$C_k = \{ \vec{x} \mid \phi_c(\vec{x}) = k \}.$$  

To achieve optimal clustering, the set of centroids and the corresponding clusters’ weights together must minimize the sum of squared weighted distances from all the data points to
their respective centroid, which is

\[
\sum_{i=1}^{N} L_{2,w_{\phi_{c}(x_i)}}^2(\vec{x}_i, \vec{c}_{\phi_{c}(x_i)}),
\]  
(Eq. 3.2)

subject to \( \forall k \prod_{j=1}^{m} w_{kj} = 1 \).

Our formulation differs from Locally Adaptive Clustering (LAC) [61]. In LAC, the constraint is the sum of weights to be one, which can lead to a trivial solution: the dimension along which the data exhibit the smallest variation is weighted one and the other dimensions receive zero weights. Thus, a regulation term representing the negative entropy of weights is added to the objective function with a coefficient. Consequently, the clustering objective is a weighted sum of vector quantization error and the regulation term. However, the critical coefficient greatly affects the quality of clustering outputs in practice, and there does not exist a simple and principal way to determine its value in LAC. In our proposal, we use the constraint that the product of the weights of any cluster must be equal to 1. This design is not trapped with the above mentioned trivial solution, and the regulation term is avoided. We do not need any user-specified parameters to control the locally weighting scheme. Note that the Euclidean distance is a special weighted distance measurement with all the weights being 1 and therefore the constraint conditions are satisfied. Our constrained minimization problem can be solved using the \textit{Lagrange Multipliers}. We state major conclusions below:

\textbf{Theorem 3.1} \textit{For the problem defined in Eq. 3.2, the optimal cluster centroids are, for } 1 \leq k \leq K, 1 \leq j \leq m,

\[ c_{kj} = \frac{1}{|C_k|} \sum_{\vec{x} \in C_k} x_j, \]  
(Eq. 3.3)

\textit{and the optimal weights are},

\[ w_{kj} = \frac{\lambda_k}{\sum_{\vec{x} \in C_k} |x_j - c_{kj}|^2}, \]  
(Eq. 3.4)

34
in which $\lambda_k = \left( \prod_{j=1}^{m} (\sum_{\bar{x} \in C_k} |x_j - c_{kj}|^2) \right)^{\frac{1}{m}}$.

**Proof:** See Section 3.7.1.

It is highly desired that Eq. 3.3 and Eq. 3.4 are the closed-form formulae so that the centroids and weights can be computed fairly efficiently during the clustering iterations. It is also interesting to see that in our scheme, the centroid of a cluster is still the center of all the points in the cluster irrespective of the different weights. As Eq. 3.4 shows, the local weighting coefficients of a cluster are non-negative and completely determined by all the points it encloses and are not directly affected by other clusters. Specifically, the component $w_{kj}$ is inversely proportional to the variance of the values in the $j$th dimension of all data points in $C_k$. If the points in the $k$th cluster differ greatly in dimension $j$, the weight $w_{kj}$ is smaller. On the other hand, if the points exhibit a strong correlation in the $j$th dimension, then a larger weight is assigned to this dimension. In general, the adaptive weights can characterize the shapes of the clusters and are expected to well reflect the heterogeneous natures of different clusters. Our formulation is intuitive and has a stable performance with no tuning.

It is possible that for some cluster $k$ and some dimension $j$, the value $\sum_{\bar{x} \in C_k} |x_j - c_{kj}|^2$ can be very small and even zero, which can cause troubles in computing the weights of this cluster. To circumvent this problem, we set a threshold in practice and when the value $\sum_{\bar{x} \in C_k} |x_j - c_{kj}|^2$ falls below this threshold, we use the threshold instead in the subsequent computations of $w_{kj}$ (in the experiments of this study, the threshold is $10^{-6}$). On the other hand, if the values $\sum_{\bar{x} \in C_k} |x_j - c_{kj}|^2$ are very large for some dimensions, it is likely that the direct computation of $\lambda_k$ could result in an overflow. Eq. 3.4 to compute weights can be rewritten in logarithm to avoid this problem, as below:

$$\log w_{kj} = \frac{1}{m} \sum_{i=1}^{m} \log (\sum_{\bar{x} \in C_k} |x_i - c_{ki}|^2) - \log (\sum_{\bar{x} \in C_k} |x_j - c_{kj}|^2).$$
Algorithm 3.1 Locally Weighted Clustering (LWC)

**Require:** a dataset of $N$ points $\mathbf{x}_i \in \mathbb{R}^m$, the number of clusters $K$.

**Ensure:** $K$ cluster centroids $\mathbf{c}_k$ and weights $\mathbf{w}_k$.

1: Start with $K$ initial centroids and set all the weights to be 1, i.e., $w_{kj} = 1$ for $1 \leq k \leq K, 1 \leq j \leq m$.

2: E-Step: Compute the membership decision $\phi_k(\mathbf{x}_i)$ for all the $N$ data points according to Eq. 3.1 and derive $K$ cluster sets $C_k$.

3: M-Step: For each cluster, recompute the centroid $\mathbf{c}_k$ with regard to all the points it has, according to Eq. 3.3 and then update the weights $\mathbf{w}_k$ according to Eq. 3.4.

4: Repeat steps 2 and 3 until converge.

The adaptiveness of locally weighted clustering can be further extended by considering (the inverse of) the covariance matrix of each individual cluster in computing the Mahalanobis distance, which can describe any arbitrarily oriented ellipsoid centered at the centroid. However, as pointed out in [132], it is not robust when a small number of data points are used to compute the covariance matrix. During the clustering process, some intermediate clusters may only have several points and the estimated ill-conditioned covariance matrix can potentially compromise the clustering accuracy. Therefore, in this study, we fit the shapes of the clusters to be ellipsoids aligned with the axes for stable performance.

Similar to K-Means, we propose an iterative procedure to reach a good partition for a given dataset, as shown in Algorithm 3.1. In the initial phase, we can use either Forgy initialization or subset furthest first for the centroid selection [73]. At the beginning, we assume that the shape of each cluster is a sphere and therefore all the weights are set to 1, indicating that the Euclidean distance is used. After the initialization, the whole procedure alternates between cluster assignments (E-step) and the updates of the centroids and the weights for individual clusters (M-step). In the E-step, each point is assigned to the closest cluster based on the local distance metric, and therefore the objective function defined in Eq. 3.2 for the new assignments surely becomes smaller. In the M-step, the centroids and the weights of the clusters are re-estimated using all the points which now belong to them, and this also certainly reduces the objective function, which has been proved in Theorem
3.1. There are a finite number of partitions dividing $N$ points into $K$ sets, and the objective function keeps decreasing from iteration to iteration. Therefore Algorithm 3.1 guarantees to converge and the converged $\vec{c}_k$ and $\vec{w}_k$ give a local minimum of the objective function (the detailed proof is available in Section 3.7.2). In practice, our algorithm LWC stops if either the data placements are stable or the user-specified maximum number of iterations is reached.

### 3.4 Clustering Under Constraints

The binary relation $R_g$ is defined for a given membership function $\phi_g$ as in Eq. 2.3. For a dataset of $N$ points, there are $\frac{(N-1) \cdot N}{2}$ unique pairs of relations in $R_g$ between different points. As pointed out by Wagstaff et al. [127,128], a small part of the relation $R_g$ is usually accessible in the clustering practice and they are naturally represented as instance-level constraints. That is, there are a certain number of pairs in the constraint set $C$ and we know $R_g(\vec{x}_i, \vec{x}_j)$ for all the pairs in $C$. If $R_g(\vec{x}_i, \vec{x}_j) = 1$, these two points must belong to the same cluster and this is called a Must-Link constraint. Otherwise, it is a Cannot-Link constraint. It is desired to have the clustering outputs satisfying these pairwise instance-level constraints. It has been shown that this partial information is fairly useful to improve the clustering accuracy and the semi-supervised clustering under constraints is a promising research direction. One example is the Constrained K-Means [128], in which each data point is individually placed in its ‘closest feasible’ cluster in the assignment phase. This motivates us to integrate our locally weighted clustering scheme with the constraints-driven clustering process.

#### 3.4.1 Chunklet Assignment Basics

Aharon et al. [22] defined a chunklet as ‘a subset of points that are known to belong to the same although unknown class’. Note that for a given set of pairwise constraints, it is
possible to combine them to form chunklets based on the transitive closure of the must-link constraints. For instance, if $R_g(\vec{x}_1, \vec{x}_2) = 1$ and $R_g(\vec{x}_2, \vec{x}_3) = 1$, then $R_g(\vec{x}_1, \vec{x}_3) = 1$ can be inferred and a chunklet can be formed by including these three points: $\Delta = \{\vec{x}_1, \vec{x}_2, \vec{x}_3\}$, whose size is the number of data points in the set, i.e., $s(\Delta) = 3$. The other type of the constraints, cannot-link, defines the relationships among different chunklets. Suppose, besides $\Delta$, there is another chunklet $\Delta' = \{\vec{x}_4, \vec{x}_5\}$. Given that $R_g(\vec{x}_3, \vec{x}_4) = 0$, then it can be inferred that chunklets $\Delta$ and $\Delta'$ should not be placed in the same cluster. Consequently, given a set of instance-level constraints, we can derive a set of chunklets and their relationships.

The conventional clustering procedures assign data points to clusters in one-by-one fashion. Given a chunklet, we can now consider assigning the points in the chunklet in bulk. Moreover, if we know two chunklets should not be in the same cluster, then their membership decisions are indeed related and we can also consider placing them at the same time. This is the basic idea of our chunklet assignment strategy, and how we decide the memberships of the chunklets are explored in detail:

For an isolated chunklet $\Delta$, which does not have any cannot-link constraints with any other chunklets, all points in $\Delta$ are assigned to the cluster which minimizes the sum of squared distances between all the points in $\Delta$ and the centroid $\vec{c}_i$:

$$\sum_{\vec{x} \in \Delta} L_{2,\vec{w}_i}^2(\vec{x}, \vec{c}_i). \quad \text{(Eq. 3.5)}$$

When there are two neighboring chunklets $\Delta$ and $\Delta'$ and there are cannot-link constraints between them, then they have to belong to different clusters. We assign $\Delta$ to cluster $i$ and $\Delta'$ to cluster $j, (i \neq j)$, in order to minimize the objective:

$$\sum_{\vec{x} \in \Delta} L_{2,\vec{w}_i}^2(\vec{x}, \vec{c}_i) + \sum_{\vec{x} \in \Delta'} L_{2,\vec{w}_j}^2(\vec{x}, \vec{c}_j). \quad \text{(Eq. 3.6)}$$
In the following, we examine the theoretical background of the above strategies and in the next subsection, we discuss how the theory can be applied in practice.

Consider a simple scenario: there are two clusters $C_1$ and $C_2$ in the dataset. For cluster $C_i$, the data values in the $j$th dimension follow the normal distribution $N(\mu_{ij}, 1)$, ($1 \leq j \leq m$), which has the mean value $\mu_{ij}$ and the unit variance for simplicity, and values of different dimensions are mutually independent. Ideally, the centroids in the ground truth are $\vec{c}_1 = (\mu_{11}, \ldots, \mu_{1m})$ and $\vec{c}_2 = (\mu_{21}, \ldots, \mu_{2m})$. As the variances are 1 in all the dimensions of both clusters, the Euclidean distance, denoted as $L^2_{2,1}$, is adopted in the following analysis.

Suppose there is a chunklet $\Delta$, that belongs to cluster $i$, i.e, $\Delta \subseteq C_i$ ($1 \leq i \leq 2$). According to Eq. 3.5, $\Delta$ is assigned to cluster $j$, if for $1 \leq j, p \leq 2, j \neq p$,

$$\sum_{\vec{x} \in \Delta} L^2_{2,1}(\vec{x}, \vec{c}_j) < \sum_{\vec{x} \in \Delta} L^2_{2,1}(\vec{x}, \vec{c}_p).$$

The probability of this event is denoted as,

$$P_\Delta(j \mid i) = P(\Delta \text{ is assigned to } C_j \mid \Delta \subseteq C_i),$$

which can be computed as below.

**Theorem 3.2** For clusters $C_1, C_2$ and chunklet $\Delta$,

$$P_\Delta(1 \mid 1) = P_\Delta(2 \mid 2) = P_a(s(\Delta)),$$

$$P_\Delta(2 \mid 1) = P_\Delta(1 \mid 2) = P_a(-s(\Delta)),$$

in which $s(\Delta)$ is the number of data points in the chunklet $\Delta$ and $P_a(x)$ is defined as,

$$P_a(x) = \Phi \left( \frac{x}{2 \sqrt{|x|} \sqrt{\sum_{j=1}^m (\mu_{1j} - \mu_{2j})^2}} \right).$$

39
and \( \Phi(x) \) is the cumulative distribution function of the standard normal distribution \( N(0, 1) \), i.e.,

\[
\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left(-\frac{u^2}{2}\right)du.
\]

**Proof:** See Section 3.7.3.

The function \( \Phi(x) \) is the cumulative distribution function, that is monotonically increasing with respect to \( x \). Hence, \( P_a(x) \) is also a monotonically increasing function. The probability to assign \( \Delta \) to its true cluster is

\[
\sum_{i=1}^{2} P(\Delta \subseteq C_i)P_\Delta(i \mid i) = P_a(s(\Delta)).
\]

Similarly, we have the mistake probability \( P_a(-s(\Delta)) \). The chance of correct assignments goes up rapidly with the increase of the size of the chunklet, while that of mistake assignments decreases. In other words, if there are more data points in a chunklet, it is more likely that \( \Delta \) is assigned to its true cluster using Eq. 3.5. As there are multiple points in a chunklet and they are independent, the chance that all of them are far away from their true centroid is much smaller than the chance that any of them is far from the centroid. Note that \( \sqrt{\sum_{j=1}^{m} (\mu_{1j} - \mu_{2j})^2} \) is exactly the distance of the true centroids, i.e., \( \mathcal{L}_{2,1}(\vec{c}_1, \vec{c}_2) \). The value \( P_a(s(\Delta)) \) becomes larger as \( \vec{c}_1 \) and \( \vec{c}_2 \) have a greater distance. Therefore, if the two centroids are far away from each other, it is generally easier to distinguish these two clusters and the probability of mistake assignments is much smaller. Theorem 3.2 reflects this intuition well.

To examine the theoretical advantage of our assignment strategy, we compare the *Average Number of Correct Assignments* (ANCA) of some well-known clustering techniques. Specifically, assume each method can find the true centroids in the ground truth and we would like to count on average, how many data points in the chunklet are assigned to their respective true cluster. The conventional K-Means \([73]\) does not utilize any constraints: it determines the membership of each point individually. The probability to assign a point
$\bar{x} \in C_i$ correctly is $P_{\{x\}}(i \mid i) = P_a(1)$, because a single point itself is a chunklet sized 1. Since the assignments of data points are independent, the occurrence of correct assignments is a binomial process with $n = s(\Delta)$ and $p = P_a(1)$ [83]. Therefore, the ANCA of K-Means is

$$\sum_{i=1}^{2} P(\Delta \subseteq C_i) \left( \sum_{j=0}^{s(\Delta)} \binom{s(\Delta)}{j} (P_a(1))^j (1 - P_a(1))^{s(\Delta) - j} \right) = s(\Delta) P_a(1).$$

Another approach, Constrained K-Means [128], decides the cluster assignment for the first point in $\Delta$ and all the rest points in $\Delta$ are forced to follow this decision and assigned to the same cluster due to the must-link constraints. Therefore, the assignments of the whole chunklet are either completely right or wrong, which solely depend on the decision of the first point. The chance of the first decision being correct is $P_{\{x\}}(i \mid i)$. Hence, its ANCA is

$$\sum_{i=1}^{2} P(\Delta \subseteq C_i) \left( s(\Delta) * P_a(1) + 0 * (1 - P_a(1)) \right) = s(\Delta) P_a(1).$$

Interestingly, in the described scenario, the above two schemes have the same number of correct assignments on average. Unlike these two methods, our chunklet assignment strategy makes a joint decision for all points in $\Delta$ at once with the chance of totally correct assignments being $P_a(s(\Delta))$. Consequently our ANCA is

$$\sum_{i=1}^{2} P(\Delta \subseteq C_i) \left( s(\Delta) * P_a(s(\Delta)) + 0 * (1 - P_a(s(\Delta))) \right) = s(\Delta) P_a(s(\Delta)).$$

Because $P_a(s(\Delta))$ is far larger than $P_a(1)$, clearly our cluster assignment is superior.

Next, we consider the assignments of two chunklets $\Delta$ and $\Delta'$ with cannot-link constraints in between, which should not be placed in the same cluster. The ANCA of K-Means is $(s(\Delta) + s(\Delta'))P_a(1)$. For Constrained K-Means, the correctness of the assignments is determined by the first decision of the points in the chunklets and the ANCA is also $(s(\Delta) + s(\Delta'))P_a(1)$. Instead, we use Eq. 3.6 to decide their memberships. The two chunklets $\Delta \subseteq C_i$ and $\Delta \subseteq C_j$
are placed in two different clusters, \( C_p \) and \( C_q \), in order to minimize the aggregated distances \((1 \leq i, j, p, q \leq 2, i \neq j, p \neq q)\). This occurs with a probability,

\[
P_{\Delta,\Delta'}(p, q \mid i, j) = P(\Delta, \Delta' \text{ are respectively assigned to } C_p, C_q \mid \Delta \subseteq C_i, \Delta' \subseteq C_j),
\]

which can be computed according to the below theorem.

**Theorem 3.3** For clusters \( C_1, C_2 \) and chunklets \( \Delta, \Delta' \),

\[
P_{\Delta,\Delta'}(1, 2 \mid 1, 2) = P_{\Delta,\Delta'}(2, 1 \mid 2, 1) = P_a(s(\Delta) + s(\Delta')),
\]

\[
P_{\Delta,\Delta'}(2, 1 \mid 1, 2) = P_{\Delta,\Delta'}(1, 2 \mid 2, 1) = P_a(-s(\Delta) - s(\Delta')).
\]

**Proof:** See Section 3.7.4.

Accordingly, the ANCA of our rule in Eq. 3.6 is the biggest, which is \((s(\Delta) + s(\Delta'))P_a(s(\Delta) + s(\Delta'))\). Intuitively, when we consider the memberships of \( \Delta \) and \( \Delta' \) together, the cannot-link constraints actually reduce the search space of all possible assignments and it is much more likely that a joint decision for the two chunklets is correct. In summary, Theorems 3.2 and 3.3 indicate that it is better to group points into chunklets and do chunklet assignments with Eq. 3.5 and Eq. 3.6. When we consider the memberships of more points collectively (either one chunklet or two neighboring chunklets), it is more likely that we assign them to their true clusters.

### 3.4.2 Constrained Clustering

For a given set of pairwise constraints, our Constrained Locally Weighted Clustering (CLWC) first builds the chunklets and then the chunklet graph. Initially, each point in the constraint set is a chunklet of size 1. For every must-link constraint, we merge the chunklets containing the two points of the constraint. This procedure continues until all must-link constraints have been processed. Next, we construct the chunklet graph by representing each chunklet
as a vertex. For each cannot-link constraint, an edge is added between the two vertices whose chunklets enclose any one of the points in the constraint. Eventually, an edge in the resulting graph indicates that the chunklets of the vertices connected by the edge (neighbor chunklets in the graph) should belong to different clusters. The generated graph, denoted as $G_c$, is used to guide the cluster assignment step and this has implicit impacts on the updates of the new centroids and the weights during iterations.

In each E-step, the memberships of all data points are re-examined. For the points not participating in any constraints, they are assigned to their closest clusters as usual. The main difference is that the chunklet assignment strategy is applied for the points of all the chunklets in $G_c$. At the start of the E-step, all chunklets are unassigned (to any cluster). CLWC picks either one or two chunklets at a time and decides their memberships until all the chunklets are assigned. As there are usually a number of chunklets in $G_c$, two questions need to be answered: which chunklets should be first chosen from $G_c$ for consideration of the memberships and which clusters they should be assigned to.

According to Theorems 3.2 and 3.3, the probability of correct assignments of the two neighboring nodes $\Delta$ and $\Delta'$ is proportional to the number of data points in they two, i.e., $s(\Delta) + s(\Delta')$. This suggests that we should pick the biggest chunklets first. To make decisions for chunklet $\Delta$, it is best to combine its assignment with that of its largest unassigned neighbor $\Delta'$ if available. Only if $\Delta$ does not have any neighbors or all its neighbors have already been assigned, is the membership of this chunklet considered singly. Specifically, let $N_u(\Delta)$ denote the set of the immediate neighbor chunklets of $\Delta$ in $G_c$ which have not yet been assigned. Define the score for each unassigned chunklet,

$$score(\Delta) = \begin{cases} s(\Delta) & \text{if } N_u(\Delta) = \emptyset, \\ s(\Delta) + \max(\{s(\Delta') \mid \Delta' \in N_u(\Delta)\}) & \text{otherwise.} \end{cases}$$

The max function is used in the score computation so that a chunklet and its largest unassigned neighbor (if available) can be decided jointly, corresponding to a smallest probability
of mis-assignments. The score of $\Delta$ is the maximum number of data points that can be considered for the memberships along with $\Delta$. Hence, if chunklet $\Delta$ of the biggest score (draws are broken randomly) has undetermined neighbors, it and its largest unassigned neighbor are selected. Otherwise only $\Delta$ is chosen for the determination of its membership at this time. As chunklets are assigned to clusters in the descending order of their sizes, the assignment decisions are generally correct and more reliable.

Next, we consider the question of how to make the assignment decision. When a single chunklet $\Delta$ is in consideration, some of its neighbors may already be assigned to some clusters and therefore these clusters are blocked from accepting $\Delta$ due to the possible violations of the cannot-link constraints between $\Delta$ and its neighbors. This effectively limits the search space for the assignment of $\Delta$. Among all the remaining feasible clusters, we pick the one which has the minimum sum of squared distance between the centroid and all the points in the chunklet. If such a cluster cannot be found, a conflict is encountered: no matter which cluster the chunklet is assigned to, some constraints are surely going to be broken. As to find cluster assignments to enforce all the constraints (specifically the cannot-links) is an NP-Hard problem [55], CLWC deals with this situation by tolerating some violations and assigning $\Delta$ to its closest cluster without considering the cannot-link constraints between itself and its assigned neighbor chunklets. As observed in our experimental study, violations are indeed a rare exception.

A similar process is designed to make a joint decision for chunklet $\Delta$ and its neighbor $\Delta'$. First we find cluster candidates for $\Delta$ and $\Delta'$ respectively. Among all the feasible choices (without putting both of them in the same cluster and violating the constraints with their already assigned neighbor chunklets), we select the one that minimizes the objective in Eq. 3.6. If we fail to find a feasible assignment, this indicates that any assignments of the two chunklets will cause conflicts with some of their assigned neighbor chunklets. In this case, we ignore the decisions of all the assigned chunklets, and put $\Delta$ and $\Delta'$ in the
clusters which minimize the objective defined in Eq. 3.6. Again, constraint violations are surely incurred, however, they rarely happen in practice.

The time complexity of our chunklet assignment algorithm is competitive to that of the K-Means. The cost of each iteration of K-Means is $O(|X|Km)$ [65] in which $|X|$ is the size of the dataset, $K$ is the number of clusters and $m$ is the dimensionality. In an efficient implementation of CLWC, at the start of each iteration, the distances between each chunklet and each cluster are computed first, which are used to decide the membership of each chunklet in the subsequent process of the iteration. The worst case time complexity of the assignment procedure is still $O(|X|Km)$. In addition, our algorithm takes fewer iterations to converge compared with K-Means, as observed in the experiments.

### 3.5 Experimental Results

#### 3.5.1 Methods and Datasets

We evaluated the clustering performance of our proposals, LWC and CLWC and compared them with other state-of-the-art techniques. All the methods are listed below.

(i) K-Means [73]: K-Means using the default Euclidean distance metric.

(ii) Bisection K-Means [146]: repeatedly partition the dataset into two parts using K-Means.

(iii) PCAC [60]: K-Means over the reduced space generated by Principal Component Analysis (PCA).

(iv) LPC [147]: K-Means over the reduced space generated by Locality Preserving Projection (LPP).

(v) LAC [61]: Locally Adaptive Clustering.
(vi) LWC: The proposed Locally Weighted Clustering.


(viii) MPCK-Means [28]: involves both metric learning and constraints satisfaction.

(ix) CLWC: The proposed Constrained Locally Weighted Clustering.

We implemented those methods except for LPP and MPCK-Means, which we obtained from the authors’ web sites [9,12]. Techniques 1 through 6 are unsupervised learning ones, while the last 3 utilize instance-level constraints to guide the cluster assignment process as well as learning the distance metric. Since the optimal number of clusters $K$ for each dataset is already known, we used them in our experiments. In the case that additional tuning parameters were needed, we used the default parameters and followed the authors’ recommendations. When they were not available, we manually tuned and reported only the best performance. Extensive experiments were carried out over the datasets in Table 6.1. Most datasets were downloaded from the UCI repository [100], among which the Digits and Letter datasets were sampled by respectively extracting characters 3, 8, 9 and A, B, as in [19,28]. The Protein dataset was used in [137].

<table>
<thead>
<tr>
<th>dataset</th>
<th>$N$</th>
<th>$m$</th>
<th>$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soybean Small</td>
<td>47</td>
<td>35</td>
<td>4</td>
</tr>
<tr>
<td>Protein</td>
<td>116</td>
<td>20</td>
<td>6</td>
</tr>
<tr>
<td>Iris Plant</td>
<td>150</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Wine Recognition</td>
<td>178</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>Heart Stat Log</td>
<td>270</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>351</td>
<td>34</td>
<td>2</td>
</tr>
<tr>
<td>Balance Scale</td>
<td>625</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Wisconsin Breast Cancer</td>
<td>683</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Digits (3,8,9)</td>
<td>1008</td>
<td>16</td>
<td>3</td>
</tr>
<tr>
<td>Letter (A,B)</td>
<td>1555</td>
<td>16</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3.1: Datasets used in the experiments
3.5.2 Evaluation Metrics

We used two common metrics to evaluate the qualities of clustering outputs of different methods. The first metric is the Rand Index [107], defined in Eq. 2.4. The second metric is the Normalized Mutual Information [19,147], defined in Eq. 2.5.

The above defined metrics were used to evaluate the accuracy of the clustering algorithms in addition to the number of violated constraints for the semi-supervised ones. We will report the number of iterations our proposals take to converge compared to the efficient techniques.

3.5.3 Unsupervised Clustering Accuracy

Each of the six unsupervised clustering methods was run 100 times with different initializations over all the datasets. For LPC and PCAC, we tested them with all the possible reduced dimensionalities and recorded their best performances. Similarly we tried different $h$’s for LAC. The averaged Rand index and NMI are summarized in Table 3.2. The methods that performed the best on different datasets with regard to a particular metric are highlighted (boldface).

In general, the two evaluation metrics are quite consistent. Although no single method can outperform all the others for all the datasets, the proposed LWC is effective in many cases. According to the Rand index (or NMI), the LWC has the best performances in 4 (5 for NMI) datasets. For the other datasets, it is within 3.9% (respectively 8.8%) compared to the best method except in the sampled hand digits dataset. In addition to good overall performance, LWC does not require any parameter tuning. Thus, our method is an advanced unsupervised method for the real-world clustering problems.

3.5.4 Constrained Clustering Accuracy

To generate constraints, we adopted the methodology in [127,128]: for each constraint, two data points were randomly picked from the dataset and if both were in the same cluster in the
Table 3.2: Accuracy of unsupervised clustering algorithms.

ground truth, a must-link constraint between them was generated. Otherwise it was a cannot-
link constraint. In each dataset, totally 1000 sets of constraints of different sizes were created
(every 50 sets were of the same size), typically ranging from 50 to 1000 constraints (25 to 500
for the Soybean dataset). The semi-supervised methods, COP-KMeans, MPCK-Means and
the proposed CLWC were tested over all constraint sets, whose average performances are
reported in Table 3.3 and Figures 3.1.a to 3.1.f. Since COP-KMeans strictly enforces all the
constraints, for many datasets, it failed to produce any feasible clustering partitions (with
different initializations) when given more than 100 constraints. We therefore only report its
performances in experiments with a small number of constraints.

As shown, CLWC generally produces much better clusters compared to the other two
methods: the accuracy curves of CLWC are almost always higher than those of MPCK-Means
for the datasets. As the number of the constraints becomes larger, indicating more partial
information is used to guide the clustering process, the accuracy of both CLWC and MPCK-
Means improves consistently. Note that the performance curves of MPCK-Means may drop
when given a small number of constraints, and the performances under constraints may
be even a little worse than those without constraints for several datasets, for example, the
performance degradation in the wine dataset under around 300 constraints. This is consistent
with observations in [28], which is due to the fact that its metric learning may become biased when there is not enough information to train the metric parameters. It is interesting to observe that CLWC does not suffer this problem, having a much smoother performance with additional constraints; there are rarely noticeable ‘dips’ in the performance of CLWC.

<table>
<thead>
<tr>
<th>dataset</th>
<th>count</th>
<th>COP-KMeans</th>
<th>MPCK-Means</th>
<th>CLWC</th>
<th>count</th>
<th>MPCK-Means</th>
<th>CLWC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Rand</td>
<td>NMI</td>
<td>Rand</td>
<td>NMI</td>
<td>Rand</td>
<td>NMI</td>
</tr>
<tr>
<td>Soybean</td>
<td>25</td>
<td>0.848</td>
<td>0.734</td>
<td>0.936</td>
<td>0.881</td>
<td>0.873</td>
<td>0.790</td>
</tr>
<tr>
<td>Protein</td>
<td>50</td>
<td>0.778</td>
<td>0.382</td>
<td>0.795</td>
<td>0.431</td>
<td>0.773</td>
<td>0.390</td>
</tr>
<tr>
<td>Iris</td>
<td>50</td>
<td>0.861</td>
<td>0.723</td>
<td>0.912</td>
<td>0.804</td>
<td>0.937</td>
<td>0.856</td>
</tr>
<tr>
<td>Wine</td>
<td>50</td>
<td>0.714</td>
<td>0.388</td>
<td>0.919</td>
<td>0.793</td>
<td>0.924</td>
<td>0.821</td>
</tr>
<tr>
<td>Heart</td>
<td>100</td>
<td>0.525</td>
<td>0.020</td>
<td>0.586</td>
<td>0.126</td>
<td>0.802</td>
<td>0.500</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>100</td>
<td>0.556</td>
<td>0.085</td>
<td>0.592</td>
<td>0.148</td>
<td>0.594</td>
<td>0.216</td>
</tr>
<tr>
<td>Balance</td>
<td>100</td>
<td>0.604</td>
<td>0.160</td>
<td>0.593</td>
<td>0.134</td>
<td>0.598</td>
<td>0.147</td>
</tr>
<tr>
<td>Breast</td>
<td>100</td>
<td>0.904</td>
<td>0.702</td>
<td>0.908</td>
<td>0.713</td>
<td>0.934</td>
<td>0.781</td>
</tr>
<tr>
<td>Digits(3,8,9)</td>
<td>150</td>
<td>0.877</td>
<td>0.730</td>
<td>0.773</td>
<td>0.651</td>
<td>0.790</td>
<td>0.658</td>
</tr>
<tr>
<td>Letter(A,B)</td>
<td>200</td>
<td>0.848</td>
<td>0.606</td>
<td>0.854</td>
<td>0.626</td>
<td>0.900</td>
<td>0.740</td>
</tr>
</tbody>
</table>

Table 3.3: Accuracy of semi-supervised clustering algorithms.

Although our constrained clustering algorithm does not guarantee the satisfaction of all constraints, only a small number of constraints were observed broken by our method in the experiments. The average numbers of violated constraints for the datasets are shown in Figures 3.2.a - 3.2.c: they grow slowly as the number of pairwise constraints increases and are much smaller compared to those of MPCK-Means.

3.5.5 Clustering Efficiency

We summarize the average number of iterations the clustering algorithms took to reach convergence in Table 3.4. The 4th and 7th columns are the numbers of the constraints in use. Compared with K-Means, which is an efficient algorithm [82], the LWC algorithm converges fairly quickly and it took a comparable number of iterations to generate the clusters. The
Figure 3.1: Accuracy of semi-supervised clustering algorithms.
CLWC algorithm generally took even fewer iterations to converge than K-Means and MPCK-Means, and the more constraints were given, the faster CLWC completed the data partition. Therefore, our proposals are also quite efficient.

### 3.6 Summary

We proposed to use local weighting vectors in order to capture the heterogeneous structures of data clusters in the feature space. Each set of weights defines the subspace spanned by
the corresponding cluster. We integrated the constrained learning into our locally weighted clustering algorithm. A set of chunklets are built upon constraints, whose points are assigned to clusters collectively. Theoretical analysis and experiments have confirmed the superiority of our new proposals.

Currently, we are investigating the proposed technique for different application domains. In particular, we have implemented a content-based image retrieval system [48]. We are also studying other approaches such as nonnegative matrix factorization and random walk techniques for constrained locally weighted clustering.

### 3.7 Proofs

#### 3.7.1 Optimal Cluster Centroids and Weights

To solve the optimization problem, we use Lagrange Multipliers. Define:

\[
F = \sum_{i=1}^{N} L^2(x_i, \bar{c}_{\phi_i(\bar{x}_i)}) - \sum_{k=1}^{K} \lambda_k (\prod_{j=1}^{m} w_{kj} - 1) = \sum_{k=1}^{K} \sum_{x \in C_k} \sum_{j=1}^{m} w_{kj} |x_j - c_{kj}|^2 - \sum_{k=1}^{K} \lambda_k (\prod_{j=1}^{m} w_{kj} - 1).
\]
For all $1 \leq k \leq K, 1 \leq j \leq m$, let
\[ \frac{\partial F}{\partial c_{kj}} = \sum_{\bar{x} \in C_k} 2w_{kj}(c_{kj} - x_j) = 0. \]

As $w_{kj} \neq 0$, then we get,
\[ c_{kj} = \frac{1}{|C_k|} \sum_{\bar{x} \in C_k} x_j. \]

Similarly, let
\[ \frac{\partial F}{\partial w_{kj}} = \sum_{\bar{x} \in C_k} |x_j - c_{kj}|^2 - \lambda_k \prod_{j' = 1, j' \neq j}^m w_{kj'} = \sum_{\bar{x} \in C_k} |x_j - c_{kj}|^2 - \frac{\lambda_k}{w_{kj}} = 0. \]

Then we have,
\[ w_{kj} = \frac{\lambda_k}{\sum_{\bar{x} \in C_k} |x_j - c_{kj}|^2}. \]

As for $1 \leq k \leq K, \prod_{j=1}^m w_{kj} = 1$, we have
\[ \lambda_k = \left( \prod_{j=1}^m \left( \sum_{\bar{x} \in C_k} |x_j - c_{kj}|^2 \right) \right)^{\frac{1}{m}}. \]

The second order partial derivatives of $F$ are computed as:
\[
\begin{bmatrix}
\frac{\partial^2 F}{\partial c_{kj}^2} & \frac{\partial^2 F}{\partial c_{kj} \partial w_{kj}} \\
\frac{\partial^2 F}{\partial w_{kj} \partial c_{kj}} & \frac{\partial^2 F}{\partial w_{kj}^2}
\end{bmatrix}
= \begin{bmatrix}
2 \sum_{\bar{x} \in C_k} w_{kj} & \sum_{\bar{x} \in C_k} 2(c_{kj} - x_j) \\
\sum_{\bar{x} \in C_k} 2(c_{kj} - x_j) & \frac{\lambda_k}{w_{kj}^2}
\end{bmatrix}.
\]

Its determinant is positive at the derived optimal weights and centroids, and therefore, they represent a minimum.
3.7.2 Convergence of Locally Weighted Clustering

Corollary 3.1 The Locally Weighted Clustering Algorithm (Algorithm 3.1) converges to a local minimum of the objective function defined in Eq. 3.2.

Proof: The objective function \( f \) is defined for the given assignments \( \phi \) and centroids \( \vec{c} \) and weights \( \vec{w} \):

\[
f(\phi, \vec{c}, \vec{w}) = \sum_{i=1}^{N} L_{2, \vec{w}_{\phi(x_i)}}^2(\vec{x}_i, \vec{c}_{\phi(x_i)}).
\]

Algorithm 3.1 starts from an initial assignment and runs from iteration to iteration. Each iteration consists of two steps: to determine cluster assignments (E-step, Line 2 in Algorithm 3.1) and to compute centroids and weights for individual clusters (M-step, Line 3).

Formally, let \( \vec{c}_i, \vec{w}^i \) and \( \phi_c^i \) respectively denote the centroids, weights, assignments derived in the \( i \)th iteration. \( \vec{c}^0 \) and \( \vec{w}^0 \) are the initial configuration, while in the algorithm \( \phi_c^0 \) is not initialized and can be any assignment. In \( \phi_c^i \), each point \( \vec{x}_i \) is assigned to its closest cluster according to weights and centroids in the last iteration, \( \vec{c}^{i-1} \) and \( \vec{w}^{i-1} \). Therefore, each E-step reduces the objective value, i.e.,

\[
f(\phi_c^i, \vec{c}^{i-1}, \vec{w}^{i-1}) \leq f(\phi_c^{i-1}, \vec{c}^{i-1}, \vec{w}^{i-1}).
\]

In each M-step, for the given \( \phi_c^i \), the optimal \( \vec{c} \) and \( \vec{w} \) are computed using Eq. 3.3 and Eq. 3.4 (as in Section 3.7.1). Hence, each M-step reduces the objective value, i.e.,

\[
f(\phi_c^i, \vec{c}, \vec{w}^{i}) \leq f(\phi_c^{i-1}, \vec{c}^{i-1}, \vec{w}^{i-1}).
\]

Overall, we have \( f(\phi_c^i, \vec{c}, \vec{w}) \) no greater than \( f(\phi_c^{i-1}, \vec{c}^{i-1}, \vec{w}^{i-1}) \). It is guaranteed that Algorithm 3.1 reduces the objective value in iterations.
The clustering problem is to group \( N \) points into \( K \) disjoint sets and there are only a finite number of data partitions. For a given \( \phi_c \), the minimal objective value is determined for the corresponding optimal centroids and weights. Therefore, the objective value for a given assignment is lower-bounded. The objective value in Algorithm 3.1 decreases gradually until the value reaches a fixed point. This fixed point is a local minimal of \( f(\phi, \vec{c}, \vec{w}) \).

3.7.3 Error Probability of One Chunklet

There are \( K \) clusters, \( C_1, C_2, \ldots, C_K \). For cluster \( C_i \), the data values in the \( j \)th dimension follow the normal distribution \( N(\mu_{ij}, 1) \).

For a chunklet \( \Delta \) that belongs to cluster \( s \) in the ground truth, \( (\Delta \subseteq C_s) \), the conditional probability that the sum of distances from points in \( \Delta \) to cluster \( i \) is smaller than that to cluster \( p \), \( (i \neq p) \), is denoted as,

\[
P_{d,1}(i, p \mid s) = P\left( \sum_{\vec{x} \in \Delta} \mathcal{L}_{2,1}(\vec{x}, \vec{c}_i) < \sum_{\vec{x} \in \Delta} \mathcal{L}_{2,1}(\vec{x}, \vec{c}_p) \mid \Delta \subseteq C_s \right).
\]

**Theorem 3.4** For \( 1 \leq i, p, s \leq K \), \( i \neq p \) we have

\[
P_{d,1}(i, p \mid s) = \Phi\left( -\frac{s(\Delta) \sum_{r=1}^{m} (\mu_{pr} - \mu_{ir}) \mu_{sr} + \frac{1}{2} s(\Delta) \sum_{r=1}^{m} (\mu_{pr}^2 - \mu_{ir}^2)}{s(\Delta) \sum_{r=1}^{m} (\mu_{pr} - \mu_{ir})^2} \right).
\]

**Proof:** We can rewrite the left hand side (LHS) as below,

\[
LHS = P\left( \sum_{\vec{x} \in \Delta} \sum_{r=1}^{m} ((x_r - \mu_{ir})^2 - (x_r - \mu_{pr})^2) < 0 \mid \Delta \subseteq C_s \right)
\]

\[
= P\left( \sum_{\vec{x} \in \Delta} \sum_{r=1}^{m} (\mu_{pr} - \mu_{ir}) x_r < \frac{1}{2} s(\Delta) \sum_{r=1}^{m} (\mu_{pr}^2 - \mu_{ir}^2) \mid \Delta \subseteq C_s \right).
\]

As \( x_r \) follows \( N(\mu_{sr}, 1) \), denoted as \( x_r \sim N(\mu_{sr}, 1) \), then

\[
(\mu_{pr} - \mu_{ir}) x_r \sim N((\mu_{pr} - \mu_{ir}) \mu_{sr}, (\mu_{pr} - \mu_{ir})^2).
\]
Define $Y = \sum_{\vec{x} \in \Delta} \sum_{r=1}^{m} (\mu_{pr} - \mu_{ir}) x_r$, following a normal distribution,

$$
N(s(\Delta) \sum_{r=1}^{m} (\mu_{pr} - \mu_{ir}) \mu_{sr}, s(\Delta) \sum_{r=1}^{m} (\mu_{pr} - \mu_{ir})^2).
$$

We can normalize $Y$ into a random variable of the standard normal distribution, $Y_N \sim N(0, 1)$, i.e.,

$$
Y_N = \frac{Y - s(\Delta) \sum_{r=1}^{m} (\mu_{pr} - \mu_{ir}) \mu_{sr}}{\sqrt{s(\Delta) \sum_{r=1}^{m} (\mu_{pr} - \mu_{ir})^2}}.
$$

Therefore, we have,

$$
LHS = P\left(Y_N < \frac{-s(\Delta) \sum_{r=1}^{m} (\mu_{pr} - \mu_{ir}) \mu_{sr} + \frac{1}{2} s(\Delta) \sum_{r=1}^{m} (\mu_{pr}^2 - \mu_{ir}^2)}{\sqrt{s(\Delta) \sum_{r=1}^{m} (\mu_{pr} - \mu_{ir})^2}}\right).
$$

As $Y_N \sim N(0, 1)$, the above equation can be further rewritten using the cumulative distribution function $\Phi$ of $N(0, 1)$.

According to the definition of $P_{d,1}(i, p | s)$, for $1 \leq i, p, s \leq K$, we have

$$
P_{d,1}(i, p | s) = 1 - P_{d,1}(p, i | s).
$$

As the probability distribution function of $N(0, 1)$ is symmetric with regard to the $x = 0$, there is a special property of its cumulative function $\Phi(x)$, that is,

$$
\Phi(x) + \Phi(-x) = 1.
$$

Therefore, we have,

$$
P_{d,1}(i, p | s) = \Phi(A),
p_{d,1}(p, i | s) = \Phi(-A),
$$

56
in which

\[ A = \frac{-s(\Delta) \sum_{r=1}^{m} (\mu_{pr} - \mu_{ir}) \mu_{sr} + \frac{1}{2} s(\Delta) \sum_{r=1}^{m} (\mu_{pr}^2 - \mu_{ir}^2)}{\sqrt{s(\Delta) \sum_{r=1}^{m} (\mu_{pr} - \mu_{ir})^2}}. \]

DISCUSSIONS:

According to Theorem 3.4, the event that the sum of distances of the points in \( \Delta \subseteq C_s \) to its true cluster \( C_s \) is smaller than that to some cluster \( C_p \), occurs with the probability,

\[ P_{d,1}(s, p \mid s) = \Phi\left( \frac{\sqrt{s(\Delta)}}{2} \sqrt{\sum_{r=1}^{m} (\mu_{pr} - \mu_{sr})^2} \right). \]

In chunklet assignment, in case of two clusters \( C_1 \) and \( C_2 \), the chance to place \( \Delta \) correctly is \( P_{d,1}(1, 2 \mid 1) \) (or \( P_{d,1}(2, 1 \mid 2) \)). This is the conclusion in Theorem 3.2. In case of more than two clusters, \( \Delta \) is assigned to cluster \( i \) if cluster \( i \) is the one closest to the points in the chunklet, i.e., for all \( 1 \leq p \leq K \), and \( p \neq i \),

\[ \sum_{\vec{x} \in \Delta} L^2_{2,1}(\vec{x}, \vec{c}_i) < \sum_{\vec{x} \in \Delta} L^2_{2,1}(\vec{x}, \vec{c}_p). \]

Each of these events, \( \sum_{\vec{x} \in \Delta} L^2_{2,1}(\vec{x}, \vec{c}_i) < \sum_{\vec{x} \in \Delta} L^2_{2,1}(\vec{x}, \vec{c}_p) \), is not necessarily independent. Consider there are 3 clusters in 2-dimensional space, \( \vec{c}_1 = (1, 0), \vec{c}_2 = (2, 0), \vec{c}_3 = (3, 0) \), it is true that, for any point \( \vec{x} \), if it is closer to \( c_1 \) than \( c_2 \), then \( \vec{x} \) is also closer to \( c_1 \) than \( c_3 \). Thus, for this example,

\[ P\left( \sum_{\vec{x} \in \Delta} L^2_{2,1}(\vec{x}, \vec{c}_1) < \sum_{\vec{x} \in \Delta} L^2_{2,1}(\vec{x}, \vec{c}_2) \cap \sum_{\vec{x} \in \Delta} L^2_{2,1}(\vec{x}, \vec{c}_1) < \sum_{\vec{x} \in \Delta} L^2_{2,1}(\vec{x}, \vec{c}_3) \mid \Delta \subseteq C_1 \right) = P_{d,1}(1, 2 \mid 1) \]
\[ \neq P_{d,1}(1, 2 \mid 1)P_{d,1}(1, 3 \mid 1). \]
Although the probability to assign $\Delta$ correctly is not expressed in a closed form for more than two clusters, generally this probability is related to $P_{d,1}(s, p \mid s)$. The more data points the chunklet $\Delta$ has, the larger the positive value $\frac{\sqrt{s(\Delta)}}{2} \sqrt{\sum_{r=1}^{m} (\mu_{pr} - \mu_{sr})^2}$ is. Hence, the corresponding probability $P_{d,1}(s, p \mid s)$ is larger, and $P_{d,1}(p, s \mid s)$ is smaller. It is more likely that the points in $\Delta$ are close to the cluster they belong to, as a group. Consequently, the probability to decide the membership of $\Delta$ correctly becomes larger with the increase of the size of the chunklet, $s(\Delta)$.

### 3.7.4 Error Probability of Two Chunklets

For chunklets $\Delta \subseteq C_s$ and $\Delta' \subseteq C_t$, $(s \neq t, i \neq p \cap j \neq q)$, denote

$$P_{d,2}(i, j, p, q \mid s, t) = P\left( \sum_{\bar{x} \in \Delta} L_{2,1}^2(\bar{x}, \bar{c}_i) + \sum_{\bar{x} \in \Delta'} L_{2,1}^2(\bar{x}, \bar{c}_j) < \sum_{\bar{x} \in \Delta} L_{2,1}^2(\bar{x}, \bar{c}_p) + \sum_{\bar{x} \in \Delta'} L_{2,1}^2(\bar{x}, \bar{c}_q) \mid \Delta \subseteq C_s, \Delta' \subseteq C_t \right).$$

**Theorem 3.5** For $1 \leq i, j, p, q, s, t \leq K$, $s \neq t, i \neq p \cap j \neq q$, we have

$$P_{d,2}(i, j, p, q \mid s, t) = \Phi\left( \frac{A}{B} \right),$$

in which

$$A = -s(\Delta) \sum_{r=1}^{m} (\mu_{pr} - \mu_{ir})\mu_{sr} - s(\Delta') \sum_{r=1}^{m} (\mu_{qr} - \mu_{jr})\mu_{tr}$$

$$+ \frac{1}{2} s(\Delta) \sum_{r=1}^{m} (\mu_{pr}^2 - \mu_{ir}^2) + \frac{1}{2} s(\Delta') \sum_{r=1}^{m} (\mu_{qr}^2 - \mu_{jr}^2),$$

$$B = \sqrt{s(\Delta) \sum_{r=1}^{m} (\mu_{pr} - \mu_{ir})^2 + s(\Delta') \sum_{r=1}^{m} (\mu_{qr} - \mu_{jr})^2}.$$
Proof:  The left hand side (LHS) can be rewritten as,

\[
LHS = P \left( \sum_{\vec{x} \in \Delta} \sum_{r=1}^{m} (\mu_{pr} - \mu_{ir})x_r + \sum_{\vec{x} \in \Delta'} \sum_{r=1}^{m} (\mu_{qr} - \mu_{jr})x_r < \right.
\]

\[
\frac{1}{2} \left( \sum_{\vec{x} \in \Delta} \sum_{r=1}^{m} (\mu_{pr}^2 - \mu_{ir}^2) + \sum_{\vec{x} \in \Delta'} \sum_{r=1}^{m} (\mu_{qr}^2 - \mu_{jr}^2) \right) | \Delta \subseteq C_s, \Delta' \subseteq C_t
\]

Define \( Y = \sum_{\vec{x} \in \Delta} \sum_r (\mu_{pr} - \mu_{sr})x_r + \sum_{\vec{x} \in \Delta'} \sum_r (\mu_{qr} - \mu_{jr})x_r \), that follows a normal distribution with the mean

\[
s(\Delta) \sum_{r=1}^{m} (\mu_{pr} - \mu_{sr})\mu_{sr} + s(\Delta') \sum_{r=1}^{m} (\mu_{qr} - \mu_{jr})\mu_{tr},
\]

and the variance

\[
s(\Delta) \sum_{r=1}^{m} (\mu_{pr} - \mu_{sr})^2 + s(\Delta') \sum_{r=1}^{m} (\mu_{qr} - \mu_{jr})^2.
\]

\( Y \) can be normalized, \( Y_N \sim N(0, 1) \), and we can derive the result of this theorem in the similar process in Theorem 3.4.

According to the definition, for \( 1 \leq i, j, p, q, s, t \leq K \), we also have,

\[
P_{d,2}(i, j, p, q \mid s, t) = 1 - P_{d,2}(p, q, i, j \mid s, t).
\]

DISCUSSIONS:

According to Theorem 3.5, we have,

\[
P_{d,2}(s, t, p, q \mid s, t) = \Phi \left( \frac{1}{2} \sqrt{s(\Delta) \sum_{r=1}^{m} (\mu_{pr} - \mu_{sr})^2 + s(\Delta') \sum_{r=1}^{m} (\mu_{qr} - \mu_{tr})^2} \right).
\]
Without prior knowledge, if the distances among cluster centroids are the same, (i.e., for
\( i, j, L_{2,1}(\bar{c}_i, \bar{c}_j) \) is some constant), then

\[
P_{d,2}(s, t, p, q \mid s, t) = \Phi\left( \frac{\sqrt{s(\Delta) + s(\Delta')}}{2} \sqrt{\sum_{r=1}^{m} (\bar{c}_{sr} - \bar{c}_{tr})^2} \right).
\]

For two clusters \( C_1 \) and \( C_2 \), the probability to determine the memberships of \( \Delta \) and
\( \Delta' \) with no mistakes is related to \( P_{d,2}(1, 2, 2, 1 \mid 1, 2) \) and \( P_{d,2}(2, 1, 1, 2 \mid 2, 1) \). This is the
conclusion in Theorem 3.3. Similar to the analysis of one chunklet in the previous section, in
case of more than two clusters, the probability to assign \( \Delta \) and \( \Delta' \) correctly is not necessarily
equal to

\[
\prod_{p=1, q=1, p \neq s \cap q \neq t}^{K} P_{d,2}(s, t, p, q \mid s, t).
\]

In general, if the sizes of the two chunklets \( s(\Delta) + s(\Delta') \) are bigger, the value \( (s(\Delta) +
\ln(\Delta')) \sum_r (\mu_{sr} - \mu_{tr})^2 \) is larger, so is \( s(\Delta) \sum_r (\mu_{pr} - \mu_{sr})^2 + s(\Delta') \sum_r (\mu_{qr} - \mu_{tr})^2 \). Therefore,
the probability is larger that \( \Delta \) and \( \Delta' \) are closer to their true clusters rather than any other
clusters, and \( P_{d,2}(p, q, s, t \mid s, t) \) is smaller. Hence, the probability to decide the membership
of the two chunklets correctly is generally larger with more data points in \( \Delta \) and \( \Delta' \).
CHAPTER 4: SUBSPACE PROJECTION AND ITS OPTIMIZATION

4.1 Introduction

Similarity search is expected to remain an active research area as efficient similarity query processing is of fundamental importance in various applications, such as time series [103], image retrieval [66] and text retrieval [129]. In these domains the exact matching may not always be preferable in different scenarios. Moreover, data collected are usually imperfect due to the effects of sampling, digitalization, lossy compressions and transformations [66]. Efficient similarity search, therefore, is important in these applications.

The goal of similarity search is to retrieve objects considered ‘similar’ to the object of interest within some user-specified threshold. There have been many models proposed to measure the (dis)similarity between objects, e.g., Euclidean distance (also known as the $L_2$ norm) [124], general $L_p$ norm [141], and Dynamic Time Warping [88]. In this chapter, we adopt the $L_2$ metric as it is the most common model and many other metrics are based on it [70, 129]. Furthermore, it has been proved that any finite metric space can be embedded into $L_2$ norm space [32].

In general, similarity retrieval requests are issued in the form of window queries, sphere queries, or $k$-nearest neighbor (kNN) queries. Window queries can be evaluated efficiently (e.g., as with the Pyramid method [27]), while kNN queries can be implemented on top of sphere queries [115]. In this study, we focus on sphere queries in $L_2$ norm space. The objects are denoted as vectors, e.g., $\vec{a} = (a_1, \ldots, a_m)$ in $m$-dimensional Euclidean space $\mathbb{R}^m$.

Problem 1 Sphere Query: Given a set of objects $A \in \mathbb{R}^m$ and a query object $\vec{q} \in \mathbb{R}^m$, find all objects $\vec{a} \in A$ whose $L_2$ norm from $\vec{q}$ is no greater than a user-specified threshold $\epsilon$, that is $\{\vec{a} \mid \vec{a} \in A, L_2(\vec{a}, \vec{q}) \leq \epsilon\}$, in which $L_2(\vec{a}, \vec{q}) = (\sum_{i=1}^{m} (a_i - q_i)^2)^{\frac{1}{2}}$.
When \( A \) is large and \( m \) is high, Problem 1 poses a serious challenge to efficient search of qualifying objects due to the so-called \textit{curse of dimensionality} [102], which causes most indexing techniques to perform poorly [130]. To address this problem, recent techniques have been proposed to reduce dimensionalities while guaranteeing no false dismissal [70]. Among these, partition-based methods [86, 141, 124, 125] divide dimensions into pre-determined disjoint groups and transform the original vectors into feature vectors with much lower dimensionality. Besides being easy to implement, these techniques are very competitive with more complex schemes. However, as the partitioning is fixed, they are not adaptive to the various characteristics of the data in diverse applications, and therefore may perform well on some datasets but not on others.

In this study, we propose SubSpace Projection (SSP), a unified framework for the partition-based reduction techniques. In SSP, dimensions could be partitioned in any order and into groups with different sizes. We examine the effects of dimension partitioning on query performance, and show that the approximation performance can be predicted using parameters computed from the data. Following these findings, we devise a greedy algorithm to optimize the partitioning without the need to examine all possible partitions.

The rest of the chapter is organized as follows. Section 4.2 provides a survey of recent techniques and an overview of our approach. The general framework of SSP is presented in Section 4.3. Section 4.4 explores configurations of SSP and the impact of dimension partitions. We solve the problem of finding a sub-optimal partition in Section 4.5 and report the results in Section 4.6. Finally we conclude the chapter in Section 4.7.

### 4.2 Related Work

Various dimension reduction methods have been proposed in the literature. Discrete Fourier Transform (DFT) [18] reduces dimensions by truncating data sequences by keeping only the low frequency Fourier coefficients. Discrete Wavelet Transform (DWT) [38] decomposes data
sequences into the wavelet coefficients and discards those corresponding to higher resolutions. Singular Value Decomposition (SVD) [90] examines the characteristics of the dataset to find the optimal linear mapping of its data. It is well known that SVD has high computation overhead because of the eigen-decomposition and is not well suited to a dynamic database [85]. Orthogonal Locality Preserving Projection (OLPP) [35] derives a low-dimensional linear manifold embedding as the optimal approximation to the local neighborhood structures of the dataset [94], while it may neglect the global structures [46]. A recent proposal in [36] extracts principal coefficients from the data with regard to the Chebyshev polynomials.

Another popular reduction approach is to compute distances from each data point to some pre-selected reference points and in the querying phase, discard disqualifying points according to the triangle inequality rule of the metric space. For instance, OMNI-Family [68] picks a set of reference points such that they are farthest apart from each other. The distances from each data point to all the reference points are used in the filtering. Another distance-based technique, iDistance [80] builds the index based on the local structure of the data. Each data point is assigned to the closest pre-selected reference point and the distance between them is used in the pruning.

Recently, several partition-based methods were proposed, including Piecewise Aggregate Approximation (PAA) [86], Segmented Means (SMEAN) [141] and Mean-Standard deviation (MS) [124,125]. The dimensions, in their original order, are partitioned into disjoint subsets of equal size. In the Euclidean space, PAA and SMEAN are identical as they extract the mean of each vector’s portion corresponding to each subset of the dimensions, whereas MS computes both the mean and the standard deviation (the definitions of mean and standard deviation will be provided in Section 4.4). These schemes are simple, easy to implement and yet outperform more sophisticated methods [124,125]. However as their mappings are static and not adaptive to the characteristics of the datasets being indexed, their performances vary greatly for different datasets. An improvement of PAA, APCA [87], enables an adaptive representation for each individual sequence by independently partitioning the dimensions
into subsets of different sizes. Although effective in pruning power, this proposal is hard to implement in practice. APCA, however, did not consider partitioning dimensions in arbitrary orders. As shown later, this has a great effect on the query performance.

Recognizing the potentials of the partition-based approach in high dimensional search, we focus our attention to addressing the limitations of its representative techniques, namely expanding the capabilities of PAA, SMEAN and especially MS to deal with datasets with various characteristics. Specifically, we contribute the following:

(i) We propose a unified framework SSP and show that PAA, SMEAN and MS are instances of this class.

(ii) We study the performance of query evaluations under various subspace selections (i.e. dimension partitions) and show that the summation of variances of features ($sv$) and its approximation ($sc$) are tell-tale indicators of the performance.

(iii) We devise an efficient way to compute the approximate performance indicator ($sc$) without materialization of the partition and propose a greedy algorithm to efficiently derive a sub-optimal partition for a given dataset to achieve better query performance.

Our proposal can adapt well to the characteristics of datasets being indexed and remains straightforward in implementation. We evaluated the proposed technique against state-of-the-art reduction techniques in extensive experiments, which clearly demonstrate the superiority of our method.

4.3 Dimensionality Reduction

4.3.1 Preliminaries

We first review necessary terms and concepts in vector geometry. Consider vectors $\vec{a}, \vec{b} \in \mathbb{R}^m$,

(i) The dot product of $\vec{a}$ and $\vec{b}$ is $\vec{a} \cdot \vec{b} = \sum_{i=1}^{m} a_i b_i$. 

64
(ii) The length of \( \vec{a} \) is \( \rho(\vec{a}) = \mathcal{L}_2(\vec{a}, \vec{0}) = \sqrt{\vec{a} \cdot \vec{a}}. \)

(iii) The dot product can also be written in terms of the angle between \( \vec{a} \) and \( \vec{b} \): \( \vec{a} \cdot \vec{b} = \rho(\vec{a})\rho(\vec{b}) \cos \theta(\vec{a}, \vec{b}). \) As any vectors considered throughout the chapter are inside the \( m \)-dimensional cube \([0, 1]^m\), it follows that \( \theta \in [0, \frac{1}{2} \pi] \).

(iv) \( \vec{a} \) and \( \vec{b} \) are orthogonal iff \( \theta(\vec{a}, \vec{b}) = \frac{1}{2} \pi \), i.e., \( \vec{a} \cdot \vec{b} = 0. \)

4.3.2 Feature Extraction

In a nutshell, SSP consists of two steps. First we choose a partition scheme \( s \), dividing the \( m \) dimensions into \( g \) disjoint subsets \( s_i \), \( 1 \leq i \leq g \) and \( g \ll m \). Every dimension \( j \) is required to belong to one and only one subset \( s_i \), denoted as \( j \in s_i \). Each \( s_i \) spans a subspace called \( \mathbb{R}_{s_i} \), and the union of all subspaces \( \bigcup \mathbb{R}_{s_i} \) forms \( \mathbb{R}^m \). The second step is to extract features for data point \( \vec{a} \in \mathbb{R}^m \). The projection of \( \vec{a} \) onto \( \mathbb{R}_{s_i} \) is denoted as \( P_{s_i}^\vec{a} \). In SSP, two values are computed for each \( P_{s_i}^\vec{a} \) with respect to a given reference vector \( \vec{r} \). Formally,

**Definition 4.1** Given reference vector \( \vec{r} \in \mathbb{R}^m \) and a partition scheme \( s \), the feature vector \( F^\vec{a} \) of data point \( \vec{a} \) with respect to \( \vec{r} \) is

\[
F^\vec{a} = (f_{c_1}^\vec{a}, \ldots, f_{c_g}^\vec{a}, f_{s_1}^\vec{a}, \ldots, f_{s_g}^\vec{a}),
\]

in which \( f_{c_i}^\vec{a} = \rho(P_{s_i}^\vec{a}) \cos \theta(P_{s_i}^\vec{a}, P_{s_i}^\vec{r}), \) and \( f_{s_i}^\vec{a} = \rho(P_{s_i}^\vec{a}) \sin \theta(P_{s_i}^\vec{a}, P_{s_i}^\vec{r}). \)

Clearly, the feature extraction of a vector \( \vec{a} \) can be independently executed for every subspace defined in the partition scheme \( s \). In each subspace \( \mathbb{R}_{s_i} \), vector \( P_{s_i}^\vec{a} \) is projected onto the reference vector and decomposed into two new vectors. This process is illustrated in Figure 4.1 (for simplicity of notation, \( \vec{x} \) in the figure stands for \( P_{s_i}^\vec{a} \)). \( \vec{x} \) is decomposed into two subvectors: the projection of \( \vec{x} \) along the reference vector, that is \( \vec{x}_1 \) in the figure, and the vector difference of \( \vec{x} \) and \( \vec{x}_1 \), that is \( \vec{x} - \vec{x}_1, \vec{x}_2 \). The corresponding feature values \( f_{c_i}^\vec{a} \)
and $f s_i^a$ are the lengths of these two vectors. It is also noted that, for each subspace, the first subvector points to the same direction as the reference vector, while the second vector points to different directions. Therefore, our feature extraction scheme is non-linear.

![Figure 4.1: Projections of vector $\vec{x}$](image)

According to Definition 4.1, an $m$-dimensional point can be reduced to a $2g$-dimensional vector. One important criteria for dimensionality reduction is to guarantee no false dismissals [70]. We will show below our proposal preserves this critical property.

**Theorem 4.1** Consider $\vec{a}, \vec{b}, \vec{r} \in \mathbb{R}^m$, for $1 \leq i \leq g$,

$$
\mathcal{L}_2(P_{s_i}^a, P_{s_i}^b) \geq |f c_i^a - f c_i^b|, \text{ and } \mathcal{L}_2(P_{s_i}^a, P_{s_i}^b) \geq |f s_i^a - f s_i^b| \tag{Eq. 4.1}
$$

$$
\mathcal{L}_2(P_{s_i}^a, P_{s_i}^b) \geq \sqrt{(f c_i^a - f c_i^b)^2 + (f s_i^a - f s_i^b)^2}. \tag{Eq. 4.2}
$$

**Proof:** See Section 4.8.1.

Summing up the results of Eq. 4.2 for all subspaces, we have the no-false-dismissal guarantee for the entire data space $\mathbb{R}^m$.

**Corollary 4.1** Given $\vec{a}, \vec{b}, \vec{r} \in \mathbb{R}^m$, then $\mathcal{L}_2(\vec{a}, \vec{b}) \geq \mathcal{L}_2(\vec{F}, \vec{F})$.

Corollary 4.1 states that the Euclidean distance between any two vectors in $\mathbb{R}^m$ is no smaller than that of their corresponding feature vectors. The most significant implication of this result is that we can build a fast no-false-dismissal search algorithm, as follows. To retrieve all vectors $\vec{a}$ inside the sphere centered at $\vec{q}$ with radius $\epsilon$ (that is, $\mathcal{L}_2(\vec{a}, \vec{q}) \leq \epsilon$), we first perform a quick-and-dirty search on the feature vectors of the original data based
on $L_2(F\vec{a}, F\vec{q}) \leq \epsilon$. The quick-and-dirty search always returns a super set of the retrieval set, and therefore the whole process guarantees to return the correct results after the final refine procedure, i.e., we will obtain the exact results as we could with an exhaustive linear scan over the original data. The detailed query processing algorithm is discussed in the next subsection.

4.3.3 Query Processing

From Definition 4.1, given a dimension partition scheme and a reference vector, an $m$-dimensional data point can be transformed into its $2g$-dimensional feature vector in linear time $O(m)$. The transformed points can be stored in a flat file or indexed by a spatial access method such as R*-tree [25]. For SSP, the dimension partition scheme and the reference vector also need to be stored. During the query processing, query point $\vec{q}$ is transformed into $F\vec{q}$ in the same fashion as the construction of the index and the filter-and-refine strategy based on Theorem 4.1 and Corollary 4.1 is adopted, as laid out in Algorithm 4.1.

Algorithm 4.1 SphereQuery

**Require:** index $T$, query vector $\vec{q}$, radius $\epsilon$

**Ensure:** a set of points whose distance from $\vec{q}$ is within $\epsilon$.

1: Compute feature vector $F\vec{q}$ for $\vec{q}$.

2: Retrieve point $\vec{a}$ if for $1 \leq i \leq g$,

$$|f_{\vec{a}_i} - f_{\vec{q}_i}| \leq \epsilon \text{ and } |f_{\vec{a}_i} - f_{\vec{q}_i}| \leq \epsilon.$$

3: For each retrieved point $\vec{a}$, keep it only if

$$\sqrt{\sum_{i=1}^{g} ((f_{\vec{a}_i} - f_{\vec{q}_i})^2 + (f_{\vec{a}_i} - f_{\vec{q}_i})^2) \leq \epsilon}.$$

4: Apply $L_2(\vec{a}, \vec{q}) \leq \epsilon$ to refine the final results.

Algorithm 4.1 consists of two filtering phases: a window query of range $\epsilon$ centered at $F\vec{q}$ (step 2) and a sphere query which discards vectors whose feature is not within $\epsilon$ from $F\vec{q}$.
(step 3). The last step 4 performs the exact distance computation for each candidate point in order to filter out all false hits. For the flat file index and tree-based structures with only window-query support, two phases filtering can be directly applied to discard disqualifying points and avoid computations as early and as much as possible. For other tree-based index implementations with support of sphere queries [25], the phase 1 filtering can be skipped.

The above procedure can be integrated into the multi-step algorithm [115,80] to support kNN query retrieval, which uses incremental similarity ranking and a priority queue to keep the top \( k \) nearest points within the current candidate set. Interested readers are referred to those works for detailed discussions.

### 4.4 Configurations of SSP

The feature extraction of SSP requires a reference vector and a partition scheme. In the following, we will fix on one particular choice of the reference vector for our detailed analysis. After that we will explore the issues of choosing dimension partition.

#### 4.4.1 Diagonal Reference Vector

First we introduce the notations of the mean and standard deviation for a given set of samples \( x_{i} = \{x_{i}|1 \leq i \leq n\} \). The mean of \( x_{i} \) is defined as:

\[
E(x_{i}) = E(\{x_{i}|1 \leq i \leq n\}) = \frac{1}{n} \sum_{i=1}^{n} x_{i}.
\]

The variance and the standard deviation of the samples are, respectively:

\[
var(x_{s}) = E(x_{s}^{2}) - (E(x_{s}))^{2} = \frac{1}{n} \sum_{i=1}^{n} x_{i}^{2} - \left(\frac{1}{n} \sum_{i=1}^{n} x_{i}\right)^{2}, \text{ and } std(x_{s}) = \sqrt{var(x_{s})}.
\]
Consider a partition scheme dividing \( m \) dimensions into \( g \) subsets, each having \( l_i \) dimensions and constituting \( l_i \)-dimensional subspace \( \mathbb{R}_{s_i} \). Let us consider a choice of the reference vector.

Recall that the feature value \( f_{c_i} \) of \( \vec{a} \) is proportional to the dot product of \( P_{s_i} \vec{a} \) and \( P_{s_i} \vec{r} \), see Section 4.3. \( P_{s_i} \vec{r} \) can be viewed as weights and the dot product \( f_{c_i} \) as a weighted sum of the components of \( P_{s_i} \vec{a} \). As such, a straightforward option is to assign all dimensions an equal weight, \( r_i = 1 \), for \( 1 \leq i \leq m \). The reference vector \( \vec{r} \) then corresponds to the diagonal of \( \mathbb{R}^m \) and \( P_{s_i} \vec{r} \) is the diagonal of subspace \( \mathbb{R}_{s_i} \). Consequently, \( f_{c_i} = \rho(P_{s_i} \vec{a}) \cos \theta(P_{s_i} \vec{a}, P_{s_i} \vec{r}) = \frac{1}{\sqrt{l_i}} \sum_{j \in s_i} a_j = \sqrt{l_i} E(\{a_j | j \in s_i\}) = \sqrt{l_i} E(P_{s_i} \vec{a}) \), and

\[
\begin{align*}
  f_{s_i} &= \sqrt{\rho^2(P_{s_i} \vec{a}) - (f_{c_i})^2} = \sqrt{l_i} \sqrt{\sum_{j \in s_i} a_j^2 l_i - \left( \frac{\sum_{j \in s_i} a_j}{l_i} \right)^2} = \sqrt{l_i} \text{std}(\{a_j | j \in s_i\}) = \sqrt{l_i} \text{std}(P_{s_i} \vec{a}).
\end{align*}
\]

Hence, \( f_{c_i} \) and \( f_{s_i} \) are \( \sqrt{l_i} \) times respectively the mean and standard deviation in subspace \( \mathbb{R}_{s_i} \). Their geometric meaning is extensively discussed in [124]. The diagonal has already been proved excellent as a reference vector [86, 124, 141, 125], which will be adopted as the reference vector henceforth this study. There are other works in deriving projection directions in linear dimension reduction techniques. Notably, in [71], dimensions are partitioned into ‘sub-patterns’ and the data in each sub-pattern set are projected onto the eigenvectors corresponding to largest eigenvalues. The cross-sub-pattern correlations are further explored on the data concatenated from sub-patterns [93]. A similar idea was developed for the image applications [69, 92], in which each image is treated as a matrix of features. Below, we discuss subspace selection.
4.4.2 In-Order Equal Partition

The *in-order equal partition* is the simplest way to divide \( m \) dimensions into \( g \) subsets. To simplify our discussion, we assume that \( m \) is a multiple of \( g \), so that all subspaces have the same dimensionality, i.e. for \( 1 \leq i \leq g \), \( l_i = \frac{m}{g} \). The first set of the \( \frac{m}{g} \) dimensions span the first subspace and the second set constitutes the second subspace and so on. For this configuration (the in-order equal partition with the diagonal reference vector), we can derive the following conclusion by rewriting Corollary 4.1 with the mean and standard deviation:

**Corollary 4.2** Consider vectors \( \vec{a}, \vec{b} \in \mathbb{R}^m \) where \( L_2(\vec{a}, \vec{b}) \leq \epsilon \), then the inequality holds that:

\[
\sqrt{\sum_{i=1}^{g} ((E(P_{\vec{a}_i}) - E(P_{\vec{b}_i}))^2 + (\text{std}(P_{\vec{a}_i}) - \text{std}(P_{\vec{b}_i}))^2) \leq \epsilon \sqrt{\frac{g}{m}}.}
\]

Observed that MS \([124, 125]\) is precisely this special configuration of SSP. Moreover, if only the means are extracted for each subspace, the resulting technique is essentially identical to PAA \([86]\) and SMEAN \([141]\). Through the above process, we have demonstrated that SSP is a generic framework for the partition-based dimension reduction techniques.

4.4.3 Impact of Dimension Partitions

We first introduce the metric to measure the performance of the approximation techniques: *selectivity*, i.e. the ratio of the number of candidate points after the filtering steps to the size of the dataset. In general, for each candidate point, one random IO access is needed to fetch the data and one exact distance computation is performed. The smaller the selectivity, the more effective the approximation technique. This metric is considered free of implementation bias, i.e., not affected by the specific choice or setup of the underlying index structure, and is widely used in related studies \([115, 102]\).

\(^1\)If \( m \) is not a multiple of \( g \), each of the first \((g - 1)\) subspaces is made up of \( \lfloor \frac{m}{g} \rfloor \) dimensions and all remaining dimensions span the last subspace. Another alternative is to pad zeroes at the end of each vector \([141]\).
The dimension partition scheme greatly affects the costs of query evaluation. For example, consider a set of 15,766 256-dimensional vectors (the dataset is described in Section 4.6), divided into two subsets: 15,000 data points and 766 query points. 10,000 dimension orderings were generated by random permutations of the dimensions and each was divided into 4 subsets of equal size (that is, the equal partition was applied based on the permuted dimension order). For each dimension partition, the feature space was constructed accordingly and the same set of queries were executed over various $\epsilon$'s and the average selectivity per query was computed.

Figure 4.2: Histograms on average selectivity under different query thresholds
Figures 4.2.a - 4.2.c plot the histograms of the average selectivities for different query thresholds. The selectivities vary greatly: when $\epsilon$ is 0.85, the best partition returned only 21 points from the filtering step, while 93% of the remaining produced more than 210 points, up to 843 points in the worst case. The difference becomes much more obvious with increasing $\epsilon$: the peak of histograms in Fig 4.2.a - 4.2.c moves to the high end of selectivity, indicating that more and more partitions have to examine a much larger number of data points compared to the best partition. Similar results have been observed for other datasets and under different settings. This leads to the conclusion that partition-based dimension reduction techniques are highly sensitive to the dimension partition scheme in use, and without any knowledge about the characteristics of the dataset, the in-order equal partition is not necessarily a good choice. This motivates us to relax the ‘in-order’ and ‘equal’ constraints, and pursue the optimal solution to the problem: given a dataset, find a proper partition dividing $m$ dimensions into $g$ sets so as to achieve the minimum selectivity overall.

### 4.5 Dimension Partition

#### 4.5.1 The Indicators

Consider a dataset $V$ of $n$ $m$-dimensional vectors, the $i$th vector of which is denoted as $\vec{v}_i = (v_{i1}, \ldots, v_{im})$. All values of dimension $j$ in the dataset are denoted as $v_{*j} = \{v_{ij} \mid 1 \leq i \leq n\}$. SSP defines a $2g$-dimensional feature vector $F^{\vec{v}_i}$ for $\vec{v}_i$:

$$F^{\vec{v}_i} = (f_{c1}, \ldots, f_{cg}, f_{s1}, \ldots, f_{sg}),$$

where $f_{cj} = \sqrt{\sum_{k \in s_j} E(v_{ik})}$, and $f_{sj} = \sqrt{\sum_{k \in s_j} \text{std}(v_{ik})}$.  

Our goal is to find a proper partition $s$ for the given set $V$. Since dimensionality reduction results in information loss, we should minimize this loss by obtaining features that are as discriminating as possible. Statistically speaking, the variance of a random variable indicates
how far its samples deviate from its average value. The greater the variance, the more diverse the features. In this sense, a good dimension reduction scheme should preserve the variance structure of the dataset as much as possible. Then the summation of the variances of the extracted features, $sv(s) = \sum_{i=1}^{g} (\text{var}(f_{c_{s_{i}}}) + \text{var}(f_{s_{s_{i}}}))$, can be used as an indicator for query performance. The theorem below shows that this metric directly reflects the performance of the given partition $s$.

**Theorem 4.2** The expected value of the pairwise square distances in the feature space of the dataset $V$ is twice the value of $sv$, i.e.

$$2 \sum_{i=1}^{g} (\text{var}(f_{c_{s_{i}}}) + \text{var}(f_{s_{s_{i}}})) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} L_2^2(F^{-}_{vi}, F^{-}_{vj}).$$

**Proof:** See Section 4.8.2.

The average squared distance between any two feature vectors is twice the $sv(s)$, and greater pairwise distances correspond to a larger value of $sv$. The relationship of $sv$ and the selectivity is further examined in real datasets. We executed the query processing in the same setting as discussed in Section 4.4.3. For each of the 10,000 dimension partitions, we computed one $sv$ value, and a large number of sphere queries were executed under different query thresholds and the average selectivity was collected. Therefore, for each query epsilon $\epsilon$, we have 10,000 pairs of selectivity and $sv$, each represented as one dot in Figures 4.3.a - 4.3.d. All four plots confirm the strong reverse relationship between the query selectivity and the value $sv$ for a partition. The two concave curves in Figures 4.3.a and 4.3.b exhibit the similar trend in different query $\epsilon$’s, that is, the bigger the value $sv$, the smaller the average query selectivity. For each $\epsilon$, we sorted 10,000 pairs of selectivity and $sv$ in the increasing order of $sv$ and plotted them in order in Figures 4.3.c and 4.3.d. Each dot in these two plots is corresponding to one dimension partition. On the left side of the figures, dimension partitions (dots) have smaller values of $sv$ and also larger selectivity. As we move to the right side, a dimension partition with a larger $sv$ value typically has to examine a much smaller
number of data points. The curves drop sharply toward the right end, indicating that only a very small number of partitions were effective in the query processing. All these figures clearly show that a good dimension partition is fairly desirable, otherwise the query algorithm needs to examine a lot of points. More importantly, the \( sv \) metric is the determining factor for selectivity; \( sv(s) \) can be viewed as an indicator of the quality of a dimension partition \( s \), that is, the dimension partition is effective in sphere search, its value \( sv(s) \) is large and therefore, the query selectivity is small. The correlation coefficients of selectivity and \( sv \) in row 1 of Table 4.1 are all negative, which validates this observation. Based on these findings, we now search for the partition scheme that maximizes the value of \( sv \).

### 4.5.2 Problem Reformulation

A straightforward, but expensive method to compute \( sv \) is to materialize all the feature vectors for a given dimension partition and then perform the computation exactly according to the definition. In our solution, we leverage the covariance matrix of the whole dataset \( V \) to compute an approximate value for \( sv \). Let \( C \) be the \( m \times m \) covariance matrix of \( V \) in which the coefficient \( c_{ij} \) is computed as

\[
c_{ij} = E(v_{si}v_{sj}) - E(v_{si})E(v_{sj}) = \frac{1}{n} \sum_{k=1}^{n} v_{ki}v_{kj} - \left( \frac{1}{n} \sum_{k=1}^{n} v_{ki} \right) \left( \frac{1}{n} \sum_{k=1}^{n} v_{kj} \right).
\]
4.3.a: Sel. vs $sv$ ($\epsilon = 0.50$)

4.3.b: Sel. vs $sv$ ($\epsilon = 1.50$)

4.3.c: Selectivity ($\epsilon = 0.50$)

4.3.d: Selectivity ($\epsilon = 1.50$)

4.3.e: $sc$ and $sv$

4.3.f: Sel. vs $sc$ ($\epsilon = 1.50$)

Figure 4.3: Relationships between selectivity and $sv$, $sc$
captures the extent that the values in the $i$th and $j$th dimensions vary together. It follows that, for a given partition $s$, we can rewrite $\text{var}(f_{c_{s_i}})$ based on the values of $C$ as follows:

\[
\text{var}(f_{c_{s_i}}) = \text{var}\left(\frac{1}{\sqrt{l_i}} \sum_{j \in s_i} v_{s_j}\right) = \frac{1}{l_i} (E((\sum_{j \in s_i} v_{s_j})^2) - (E(\sum_{j \in s_i} v_{s_j}))^2)
\]

\[
= \frac{1}{l_i} \sum_{j \in s_i} \sum_{k \in s_i} (E(v_{s_j}v_{s_k}) - E(v_{s_j})E(v_{s_k})) = \frac{1}{l_i} \sum_{j \in s_i} \sum_{k \in s_i} c_{jk}.
\]

Accordingly, the value $sc(s) = \sum_{i=1}^{g} \text{var}(f_{c_{s_i}})$ can now be derived efficiently. It is clear that $sc$ is part of the $sv$ metric. In the experiment described in the previous subsection, for each of the 10,000 dimension partitions, the corresponding $sv$ and $sc$ values were computed and plotted in 4.3.e. The dots (one per one partition) form a perfect straight line, indicating that $sv$ and $sc$ have a strong positive linear dependence. Their relationships to selectivity are almost identical (see Table 4.1). Figures 4.3.b and 4.3.f look highly similar with both $sc$ and $sv$ having a reverse trend with the selectivity. Therefore, $sc$ also can be seen as a performance indicator for query processing just like $sv$. The bigger the value $sc$, the smaller the query selectivity. Compared to $sv$, a significant advantage of $sc$ is that it can be computed efficiently using the covariance matrix $C$, and there is no need to materialize any feature vectors for every examined partition. The matrix $C$ can be computed once and reused to compute $sc$ for different partitions. Consequently, we reformulate our problem as follows:

*For a given dataset, find the partition $s$ of the data’s dimensions into $g$ disjoint sets so that $sc(s)$ is maximized, where*

\[
sc(s) = \sum_{i=1}^{g} \left(\frac{1}{l_i} \sum_{j \in s_i} \sum_{k \in s_i} c_{jk}\right).
\]
4.5.3 The GreedySC solution

The above optimization problem can be restricted to a 3-Partition problem, which is known to be \(NP-Complete\). To find a sub-optimal solution, we design an iterative greedy solution, \textit{GreedySC}, which repeatedly refines the partitions till a satisfied one is found.

Initially the partition \(s\) is the simplest configuration: equal partition in the original order, i.e., the \(i\)th dimension set is 
\[ s_i = \{ j \mid (i-1)\frac{m}{g} + 1 \leq j \leq i\frac{m}{g} \}, \]
and spans the \(i\)th subspace. In each iteration, a dimension set may either accept or give up a dimension in order to increase the overall score of \(sc\). Specifically, suppose in the current partition \(s\), the \(i\)th dimension is in the \(j\)th subspace. To obtain a new partition, dimension \(i\) may leave \(s_j\) and join \(s_{j'}\), the dimension set of the \(j'\)th subspace. Let the new partition be \(s_{\text{join}}(s, i, j')\), and define the matrix \(\text{join}\) as changes of the \(sc\) due to these operations, for \(1 \leq i \leq m, 1 \leq j' \leq g\), and \(i \in s_j\),

\[
\text{join}(i, j') = \begin{cases} 
sc(s_{\text{join}}(s, i, j')) - sc(s) & \text{if } j \neq j', \\
0 & \text{otherwise}.
\end{cases}
\]

There are \(m \times g\) entries in \(\text{join}\). In each greedy iteration, the \(\text{join}\) matrix is updated in accordance with the partition in the current iteration and the entry with the largest value is selected. If the selected score is positive, the corresponding join operation is performed and the new partition will be used in the next round. Otherwise, the process is terminated, as indicated in Algorithm 4.2. In other words, in each iteration, we pick a join operation that results in the largest increase of \(sc\) based on the current dimension partition until convergence.

There are a finite number of dimension partitions, thus the possible values of \(sc\) are bounded. GreedySC increases \(sc\) from iteration to iteration and this process always converges. Although GreedySC does not guarantee to produce a global optimal partition and
Algorithm 4.2 GreedySC

Require: \( n \) \( m \)-dimensional dataset \( V \), number of partitions \( g \)
Ensure: dimension partition \( s \)
1: Compute the covariance matrix \( C \) of dataset \( V \).
2: Let \( s \) be the in-order equal partition.
3: while TRUE do
4: Compute the \( \text{join} \) with respect to current partition.
5: Pick a pair \((i, j')\) such that \( \text{join}(i, j') \) is the maximum value in the \( \text{join} \) matrix.
6: if \( \text{join}(i, j') > 0 \) then
7: Let the \( i \)th dimension join the \( j' \)th subspace and update \( s \).
8: else
9: break
10: end if
11: end while

might get stuck in local optima, we shall find in the experimental section that this solution works very well for different datasets in practice.

Computing entries in the \( \text{join} \) matrix directly in each iteration is costly. We add two auxiliary data structures \( \text{rowp} \) and \( \text{squp} \) to accelerate this regular update. For \( 1 \leq i \leq m, 1 \leq j \leq g \), define

\[
\text{rowp}(i, j) = \sum_{k \in s_j} c_{ik},
\]

and for \( 1 \leq i \leq g \),

\[
\text{squp}(i) = \sum_{j \in s_i} \sum_{k \in s_i} c_{jk}.
\]

Accordingly, in the first step of the \textbf{while} loop (Line 4 in Algorithm 4.2), \( \text{rowp} \) and \( \text{squp} \) are first computed to facilitate the computation of the \( \text{join} \) matrix, which is based on the following result:
Theorem 4.3 Consider partition $s$, and suppose $i \in s_j$ where $j \neq j'$. If $l_j > 1$, then

$$join(i, j') = \left( \frac{1}{l_j - 1} - \frac{1}{l_j} \right) squp(j) + \left( \frac{1}{l_{j'} + 1} - \frac{1}{l_{j'}} \right) squp(j') - \frac{2}{l_j - 1} rowp(i, j)$$

$$+ \frac{2}{l_{j'} + 1} rowp(i, j') + \left( \frac{1}{l_j - 1} + \frac{1}{l_{j'} + 1} \right) c_{ii},$$

and if $l_j = 1$, then

$$join(i, j') = \frac{1}{l_{j'} + 1} \left( squp(j') + 2 rowp(i, j') + c_{ii} \right) - c_{ii} - \frac{1}{l_{j'}} squp(j').$$

Proof: See Section 4.8.3.

With the availability of $rowp$ and $squp$, unnecessary computations can be avoided:

(i) Cheap update of $rowp$ and $squp$: only a part of the entries in the $rowp$ and $squp$ need to be recomputed in each iteration, which can be done incrementally.

(ii) Result sharing: entries in $rowp$ and $squp$ are used to compute multiple entries in $join$.

Consequently, the time complexity of each greedy iteration has been effectively reduced from $O(gm^3)$ to $O(gm + m^2)$, thanks to the auxiliary structures.

4.6 Experimental Studies

We evaluated the performance of the proposed greedy method and compared the tuned SSP with other well-known state-of-the-art dimensionality reduction methods. All implementations were coded in C and run on a P4 computer with 1GB RAM running Ubuntu Linux 5.04. Extensive experiments were carried out on six datasets described in Table 4.2.
Table 4.2: Datasets used in the experiments

<table>
<thead>
<tr>
<th>name</th>
<th>size</th>
<th>dim</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>samp</td>
<td>15766</td>
<td>256</td>
<td>resized 16x16 images [124]</td>
</tr>
<tr>
<td>hist</td>
<td>15766</td>
<td>256</td>
<td>256-bin image histograms [124]</td>
</tr>
<tr>
<td>stock</td>
<td>6500</td>
<td>256</td>
<td>6500 companies’ stock prices [7]</td>
</tr>
<tr>
<td>sat</td>
<td>4435</td>
<td>32</td>
<td>satellite spectral data [100]</td>
</tr>
<tr>
<td>corel</td>
<td>59895</td>
<td>32</td>
<td>Corel image dataset [5]</td>
</tr>
<tr>
<td>pat</td>
<td>5000</td>
<td>128</td>
<td>synthetic data on two-pat model [15]</td>
</tr>
</tbody>
</table>

4.6.1 Selectivity, $sv$ and $sc$

The experiments on the *samp* dataset in Section 4.4.3 have confirmed the impacts of dimension partitions on query performance and justified the need for efficient algorithms to determine proper partitions for a given dataset. The analysis and experiments in Section 4.5 have established the correlations between query selectivity, $sv$, and $sc$. To further underscore these findings, we repeated the experiments on the other datasets and under many different settings.

Table 4.3 summarizes the results, with the first 5 columns describing the experimental settings. The result of one setting on the *samp* dataset is described in the first row: we examined 10,000 dimension partitions, each of which divided the 256 dimensions into 2 subsets. For each partition, we measured the selectivity of the filtering step over different query thresholds. For $\epsilon = 0.85$, the ratio of the average selectivity to the smallest selectivity over all partitions is 5.23 (column 7), indicating that the selectivity varies greatly under different partitions and the best query performance is much better than the average. Observe that the relative difference between the average selectivity and the smallest one is getting smaller for increasing $\epsilon$ (column 11) and their absolute difference is indeed quite significant because the selectivities also become larger as $\epsilon$ increases. We also recorded the correlations between the selectivity (*sel* for short), $sv$, and $sc$ (in column 8-9, 12-13). All the correlation coefficients are negative and close to -1, an indication of a strong inverse relationship. The values in the last column further confirm the highly mutual dependence of $sv$ and $sc$. Note
that we carried out the experiments under different settings such as different numbers of partitions (compares row 1-4) and from moderate to large numbers of queries (in row 5 the spatial join was performed, i.e. all data points were used as queries). Another interesting observation is that the average ratio really grows rapidly from about 5.2/2.6 to 29/14 for $\epsilon = 0.85/1.50$ (compare rows 1 through 4). This implies that when the dimensions are divided into a larger number of groups, there are more choices to partition the dimensions, but it is more likely to get poor partitions and therefore poor performance. In summary, the results of these experiments spell out the need for an effective scheme to obtain salient dimension partitions for the dimension reduction technique and confirm the roles of $sv$ and $sc$ as indicators of query performance.

### Table 4.3: Correlation of selectivity, $sv$ and $sc$

<table>
<thead>
<tr>
<th>dataset</th>
<th>data</th>
<th>query</th>
<th>partition</th>
<th>permute</th>
<th>$\epsilon$</th>
<th>ratio</th>
<th>$sel, sv$</th>
<th>$sel, sc$</th>
<th>$\epsilon$</th>
<th>ratio</th>
<th>$sel, sv$</th>
<th>$sel, sc$</th>
<th>$sv, sc$</th>
</tr>
</thead>
<tbody>
<tr>
<td>samp</td>
<td>15000</td>
<td>766</td>
<td>2</td>
<td>10000</td>
<td>0.85</td>
<td>5.23</td>
<td>-0.90</td>
<td>-0.91</td>
<td>1.50</td>
<td>2.61</td>
<td>-0.97</td>
<td>-0.98</td>
<td>0.9993</td>
</tr>
<tr>
<td>samp</td>
<td>15000</td>
<td>766</td>
<td>4</td>
<td>10000</td>
<td>0.85</td>
<td>22.14</td>
<td>-0.86</td>
<td>-0.86</td>
<td>1.50</td>
<td>5.36</td>
<td>-0.95</td>
<td>-0.95</td>
<td>0.9996</td>
</tr>
<tr>
<td>samp</td>
<td>15000</td>
<td>766</td>
<td>8</td>
<td>2000</td>
<td>0.85</td>
<td>28.61</td>
<td>-0.83</td>
<td>-0.82</td>
<td>1.50</td>
<td>11.78</td>
<td>-0.95</td>
<td>-0.95</td>
<td>0.9993</td>
</tr>
<tr>
<td>samp</td>
<td>15000</td>
<td>766</td>
<td>16</td>
<td>2000</td>
<td>0.85</td>
<td>29.03</td>
<td>-0.78</td>
<td>-0.79</td>
<td>1.50</td>
<td>13.89</td>
<td>-0.88</td>
<td>-0.89</td>
<td>0.9996</td>
</tr>
<tr>
<td>samp</td>
<td>15766</td>
<td>15766</td>
<td>4</td>
<td>2000</td>
<td>0.85</td>
<td>24.76</td>
<td>-0.84</td>
<td>-0.84</td>
<td>1.50</td>
<td>6.63</td>
<td>-0.95</td>
<td>-0.95</td>
<td>0.9997</td>
</tr>
<tr>
<td>hist</td>
<td>15000</td>
<td>766</td>
<td>4</td>
<td>10000</td>
<td>0.012</td>
<td>48.11</td>
<td>-0.78</td>
<td>-0.77</td>
<td>0.02</td>
<td>14.03</td>
<td>-0.88</td>
<td>-0.88</td>
<td>0.9872</td>
</tr>
<tr>
<td>stock</td>
<td>6100</td>
<td>400</td>
<td>4</td>
<td>10000</td>
<td>0.01</td>
<td>2.61</td>
<td>-0.93</td>
<td>-0.93</td>
<td>0.025</td>
<td>1.57</td>
<td>-0.97</td>
<td>-0.97</td>
<td>0.9996</td>
</tr>
<tr>
<td>sat</td>
<td>4200</td>
<td>235</td>
<td>2</td>
<td>10000</td>
<td>0.04</td>
<td>2.02</td>
<td>-0.49</td>
<td>-0.28</td>
<td>0.10</td>
<td>1.52</td>
<td>-0.64</td>
<td>-0.50</td>
<td>0.7867</td>
</tr>
<tr>
<td>corel</td>
<td>58000</td>
<td>1895</td>
<td>2</td>
<td>5000</td>
<td>0.06</td>
<td>1.80</td>
<td>-0.75</td>
<td>-0.69</td>
<td>0.18</td>
<td>1.61</td>
<td>-0.90</td>
<td>-0.79</td>
<td>0.8737</td>
</tr>
<tr>
<td>pat</td>
<td>4750</td>
<td>250</td>
<td>4</td>
<td>10000</td>
<td>0.60</td>
<td>12.64</td>
<td>-0.60</td>
<td>-0.75</td>
<td>0.90</td>
<td>11.64</td>
<td>-0.75</td>
<td>-0.87</td>
<td>0.9225</td>
</tr>
</tbody>
</table>

### 4.6.2 Performance of the Greedy Solutions

Our greedy solutions is designed to find a sub-optimal dimension partition for a given dataset. With the converged partition returned by GreedySC, SSP is applied to execute the sphere query, which is denoted as GS-SSP. Figure 4.4.a plots the result of a running case of GreedySC on the samp dataset with 4 partitions, which took 103 iterations to converge. As expected, the partition is continuously improved until converge, that is, from iteration to iteration, the derived partition has a greater value of $sc$ and the corresponding selectivity becomes smaller.
The initial selectivity is 6.4 times larger than the converged selectivity for $\epsilon = 1.50$. The converged partition was much better than the initial one, and GreedySC is effective.

To test the performance of the proposed greedy solution extensively, for each of the 6 datasets in Table 6.1, the order of the dimensions was shuffled randomly to generate 1,000 new datasets. Figures 4.4.b and 4.4.c show the performance of GS-SSP on the 1,000 samp datasets. The initial selectivities spread across a wide range while the converged selectivities are consistent and much smaller. GreedySC can adapt to the characteristics of the data and help make the query processing more efficient. Similar conclusions were observed for other

![Figure 4.4: SSP before and after GreedySC](image-url)
datasets. The average ratios of the initial selectivity to the converged selectivity for different datasets are summarized in Table 4.4. It is clear that GreedySC is effective in finding a sub-optimal dimension partition for a dataset, which generally only requires to examine a small number of data points. It is also interesting to note that the ratios in columns 9 and 11 of Table 4.4 are close to or even greater than those in columns 7 and 11 of Table 4.3 respectively under the same setting.

<table>
<thead>
<tr>
<th>dataset</th>
<th>experimental settings</th>
<th>efficiency</th>
<th>initial sel./converged sel.</th>
</tr>
</thead>
<tbody>
<tr>
<td>hist</td>
<td>data: 15000 query: 766 partition: 4 permute: 1000</td>
<td>step: 274 time: 6.5638 ϵ: 0.012 ratio: 37.37 ϵ: 0.02 ratio: 13.70</td>
<td></td>
</tr>
<tr>
<td>stock</td>
<td>data: 6100 query: 400 partition: 4 permute: 1000</td>
<td>step: 200 time: 2.7082 ϵ: 0.01 ratio: 2.58 ϵ: 0.025 ratio: 1.58</td>
<td></td>
</tr>
<tr>
<td>sat</td>
<td>data: 4200 query: 235 partition: 2 permute: 1000</td>
<td>step: 14 time: 0.0552 ϵ: 0.04 ratio: 1.62 ϵ: 0.10 ratio: 1.41</td>
<td></td>
</tr>
<tr>
<td>corel</td>
<td>data: 58000 query: 1895 partition: 2 permute: 1000</td>
<td>step: 16 time: 0.8941 ϵ: 0.06 ratio: 2.97 ϵ: 0.18 ratio: 2.13</td>
<td></td>
</tr>
<tr>
<td>pat</td>
<td>data: 4750 query: 250 partition: 4 permute: 1000</td>
<td>step: 99 time: 0.6319 ϵ: 0.60 ratio: 62.10 ϵ: 0.90 ratio: 50.86</td>
<td></td>
</tr>
</tbody>
</table>

Our results also help answer an intriguing question: for popular datasets, such as the ones in our experiments, how good are their default dimension orders with regard to partitioning, and how much can our method improve the retrieval performance? In some datasets such as samp, hist and stock prices, adjacent values are strongly correlated (e.g. measurements at the short-time intervals or similar attributes) so that their original-order partitions are close to the converged, sub-optimal partitions. For the others, rearranging the dimensions results in a significant improvement: initial selectivities are on average 1.5, 2.8, 3.9 times the converged selectivities respectively over a wide range of ϵ’s. Thus, our method can indeed help guarantee an improved performance in query evaluation.

In terms of resources, our greedy solution is fairly efficient. The dominating cost is the computation of the covariance matrix, rather than the greedy iterations, thanks to the auxiliary data structures, rowp and squp. For instance, for the samp dataset, GreedySC requires only 197 steps and 6.5693 seconds to converge, of which 6.5288 seconds are contributed to the computation of the covariance matrix (see columns 6 and 7 in Table 4.4) with the utilization
of the open source mathematical package IT++ 3.10.2 [2]. Without those structures, the method would take over 500 seconds to converge.

### 4.6.3 Compare with Other Techniques

We assessed the performance of the tuned SSP, GS-SSP, against the techniques discussed in Section 4.2. PAA and SMEAN are identical under the Euclidean metric. DWT with Haar basis has almost identical query performance as PAA/SMEAN if the dimensionality of the dataset is a power of 2. OLPP used the default parameters of its authors to construct the similarity matrix. The implementation of iDistance was obtained from its authors and enhanced to support sphere queries, while the other methods were built on top of R*-tree [4]. The page size for all schemes were set to be 4KB. To ensure fairness, we used the same parameters for the other 7 schemes, while the configuration of iDistance followed the suggestion of the authors (i.e. the number of reference points is twice the number of the original dimensions). Consequently, the space overheads of the techniques are comparable with each other. As iDistance is essentially a one-dimensional reduction method, iDistance was not included in the investigation of the relationship of query selectivity with the change of feature dimensions (the number of partitions is half of the feature dimensions for SSP) as shown in Figures 4.5.a - 4.5.b. Figures 4.5.c - 4.6.b plot the selectivities and the IO access counters for different techniques with various $\epsilon$’s. The results clearly show that GS-SSP consistently outperforms the others under various settings. This is attributed to its effective filtering in the feature space, thus incurring fewer points examined in the post-processing step and therefore fewer disk operations. It is interesting to observe that the performance curves for most methods possess the similar characteristics with different query thresholds except iDistance. We speculate that this could be due to the data-clustering design that iDistance relies on. iDistance divides data into clusters. During a query process, clusters that are far away from the query point get pruned, and the search is carried out locally.
Figure 4.5: Selectivity of different techniques in a small number of clusters. This is a one-dimensional reduction method and it is less effective to execute local search in a cluster. It may have to examine most of the points in this cluster even with a small query threshold. As a result, iDistance had quite a large query selectivity with a small query threshold and the number of data points to be examined slowly approaches to the number of data points in the clusters to be considered with the increase of $\epsilon$. Moreover, the performance of iDistance heavily depends on the quality of clustering.
4.6.a: samp: 16x16 images

4.6.b: Corel images

Figure 4.6: IO access counter of different techniques

4.7 Summary

We have proposed SSP, an effective dimension reduction framework and a generalization of PAA, SMEAN and MS. As the partition-based methods are not adaptive to the characteristics of different datasets, this motivated us to analyze indicators of query performance. We further designed efficient algorithms to effectively partition the space into subspaces to improve performance significantly compared to the above techniques and other state-of-the-art schemes.

4.8 Proofs

4.8.1 No False Dismissal

Proof: Consider the $i$th subspace $R_{s_i}$, and denote $P_{s_i}^x$, $P_{s_i}^y$, $P_{s_i}^z$ as $\vec{x}$, $\vec{y}$, $\vec{w}$ respectively for simplicity of notations. Then the projection of $\vec{x}$ on $\vec{w}$, $P_{||\vec{w}}^x$, is $\frac{\vec{x} \cdot \vec{w}}{\rho^2(\vec{w})} \vec{w}$. Since $\theta(\vec{x}, \vec{w}) \in [0, \frac{1}{2} \pi]$, then $\cos \theta(\vec{x}, \vec{w}) \geq 0$ and we have

$$\rho(P_{||\vec{w}}) = \rho\left(\frac{\vec{x} \cdot \vec{w}}{\rho^2(\vec{w})} \vec{w}\right) = \rho(\vec{x}) \cos \theta(\vec{x}, \vec{w}) = fc_1 \vec{a}.$$
Let \( P \vec{x} \) be \( \vec{x} - \frac{\vec{x} \cdot \vec{w}}{\rho(\vec{w})} \vec{w} \), which is a vector in the manifold perpendicular to \( \vec{w} \) [58]. It follows that \( P \vec{x} \| \vec{w} + P \vec{x} \perp \vec{w} = \vec{x} \). Since these two components \( P \vec{x} \| \vec{w} \) and \( P \vec{x} \perp \vec{w} \) are the \( \vec{w} \)-parallel and \( \vec{w} \)-perpendicular components of \( \vec{x} \) respectively, and \( \theta(\vec{x}, \vec{w}) \in [0, \frac{1}{2} \pi] \), it is true that \( \sin \theta(\vec{x}, \vec{w}) \) is no smaller than zero and we have

\[
\rho(P \vec{x} \perp \vec{w}) = \sqrt{\rho^2(\vec{x}) - \rho^2(P \vec{x} \| \vec{w})} = \rho(\vec{x}) \sin \theta(\vec{x}, \vec{w}) = f s_{S}^\vec{x}.
\]

Similar analysis can be applied to the vector \( \vec{y} \). Let us examine the illustration depicted in Fig. 4.7. The manifold \( \mathcal{H}_s \) is the hyperplane which passes through (the end point of) \( \vec{y} \) and whose normal is \( \vec{w} \). The projection of \( \vec{x} \) on \( \mathcal{H}_s \) is denoted as \( \vec{z} \). Then the vector \( (\vec{x} - \vec{z}) \) is parallel to \( \vec{w} \), and

\[
\mathcal{L}_2(\vec{x}, \vec{z}) = \rho(\vec{x} - \vec{z}) = |\rho(P \vec{x} \perp \vec{w}) - \rho(P \vec{y} \perp \vec{w})|.
\]

Consider the hyperplane \( \mathcal{H} \) passing through the origin \( \vec{o} \) and perpendicular to the \( \vec{w} \). The origin \( \vec{o} \) and the end points of \( P \vec{x} \perp \vec{w} \) and \( P \vec{y} \perp \vec{w} \) form a triangle. The triangle inequality requires that \( \mathcal{L}_2(P \vec{x} \perp \vec{w}, P \vec{y} \perp \vec{w}) \geq |\rho(P \vec{x} \perp \vec{w}) - \rho(P \vec{y} \perp \vec{w})| \). Now we prove the second part of Theorem 1. Note that \( \mathcal{L}_2(P \vec{x} \perp \vec{w}, P \vec{y} \perp \vec{w}) \) is equal to \( \mathcal{L}_2(\vec{y}, \vec{z}) \) because \( (P \vec{x} \perp \vec{w} - P \vec{y} \perp \vec{w}) \) is the projection of the \( (\vec{z} - \vec{y}) \) onto \( \mathcal{H} \). Three ending points of \( \vec{x}, \vec{y}, \vec{z} \) form a right triangle in which the line between \( \vec{x} \) and
\( \vec{y} \) is the hypotenuse. Hence

\[
\mathcal{L}_2^2(x, y) = \mathcal{L}_2^2(x, \vec{z}) + \mathcal{L}_2^2(y, \vec{z}) \geq \rho(P^2_{\parallel x} - \rho(P^y_{\parallel y}))^2 + \rho(P^2_{\perp x} - \rho(P^y_{\perp y}))^2.
\]

The above completes the proof for Eq. 4.2, which in turn implies the right side of Eq. 4.1 lower-bounds its left side in each \( \Re_s \).

4.8.2 Relationship of Indicators and Pairwise Distances

**Proof:** There are a total of \( n^2 \) pairwise distances among the vectors in the feature space. Expanding the right hand side of the above equation, we have

\[
\frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{g} \left((f_{ck} - f_{ck})^2 + (f_{sk} - f_{sk})^2\right) = \sum_{k=1}^{g} \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(f_{ck}^2 + f_{ck}^2 + f_{sk}^2 + f_{sk}^2 - 2f_{ck}f_{ck} - 2f_{sk}f_{sk}\right) = 2 \sum_{k=1}^{g} \left(E(f_{ck}^2) + E(f_{sk}^2) - (E(f_{ck}))^2 - (E(f_{sk}))^2\right) = 2 \sum_{i=1}^{g} (\text{var}(f_{ci}) + \text{var}(f_{si})).
\]

4.8.3 The Correctness of the Greedy Algorithm

**Proof:** Suppose that, initially, the \( j \)th and \( j' \)th dimension sets are \( s_j \) and \( s_{j'} \) respectively. When dimension \( i \) joins \( s_{j'} \), the resulting dimension sets are now \( s_j - \{i\} \) and \( s_{j'} \cup \{i\} \). We
have

\[
\sum_{u,v \in s_j \setminus \{i\}} c_{uv} = squp(j) - 2rowp(i, j) + c_{ii}, \quad (\text{Eq. 4.3})
\]

\[
\sum_{u,v \in s_{j'} \cup \{i\}} c_{uv} = squp(j') + 2rowp(i, j') + c_{ii}, \quad (\text{Eq. 4.4})
\]

The join operation only affects the \(j\)th and \(j'\)th subspace, thus the change in the \(sc\) score is, assuming \(l_j > 1\),

\[
\text{join}(i, j') = \frac{1}{l_j - 1} \sum_{u,v \in s_j \setminus \{i\}} c_{uv} + \frac{1}{l_j' + 1} \sum_{u,v \in s_{j'} \cup \{i\}} c_{uv} - \frac{1}{l_j} \sum_{u,v \in s_j} c_{uv} - \frac{1}{l_j'} \sum_{u,v \in s_{j'}} c_{uv}. \quad (\text{Eq. 4.5})
\]

However, if \(l_j = 1\), dimension \(i\) is the last dimension in the set \(s_j\) and \(squp(j) = c_{ii}\). If dimension \(i\) leaves \(s_j\),\(^2\) the \(j\)th set becomes empty. The change of \(sc\) is

\[
\text{join}(i, j') = \frac{1}{l_j' + 1} \sum_{u,v \in s_{j'} \cup \{i\}} c_{uv} - c_{ii} - \frac{1}{l_j'} \sum_{u,v \in s_{j'}} c_{uv}. \quad (\text{Eq. 4.6})
\]

Substituting Eq. 4.3 and Eq. 4.4 into \(\text{join}(i, j')\) (Eq. 4.5 and Eq. 4.6) produces the results to be proved.

\(^2\)Theoretically, for GreedySC, some subspaces might disappear and there are a fewer number of dimension sets after convergence. However, this rarely occurs as the diagonal entries in the covariance matrix typically are much larger than the other entries, and using more subspaces (i.e. the dimensionality of the feature space is higher) can better preserve the variance structure of the dataset. As a result, the number of subspaces in the converged partition of GreedySC is always the same as the input \(g\) in practice.
As discussed in the previous chapter, similarity search is to effectively identify objects considered ‘similar’ to the query object [66]. There are three main aspects of similarity search, namely, data representations, distance metrics and query evaluation techniques, which have attracted a lot of research attentions in recent years.

The Single Point Model (SPM) is the well established model to characterize data objects. That is, an object is represented as a vector (i.e., a single point) in the multi-dimensional space. For instance, an array of feature values (e.g., color, shape and texture) can be extracted from an image. A web page can be described by its term-frequency vector. Different distance functions have been proposed to measure the (dis)similarity between points in the data space, e.g., Euclidean distance and the generalized $L_p$ norm. Similarity retrieval requests are usually issued in form of $k$-nearest neighbor ($k$NN) query\footnote{The value $k$ is a user specified parameter.} [115], which aims to efficiently identify the top $k$ nearest objects in the database with respect to the given query object. That is, the result set contains $k$ objects whose distances to the query object are no greater than all the other objects which are not included in the result set. Many sophisticated techniques have been designed to efficiently evaluate nearest neighbor queries against a large dataset of points in the high-dimensional space, including spatial access method [25], quick-and-dirty search [115], metric indexing [80] and dimensionality reduction [124].

In this study, we are particularly interested in a new type of data representations, namely Multiple Point Model (MPM), in which an object is characterized by a set of vectors in the multi-dimensional space. In other words, an object is a point set and its size (the number of points it has) can differ from object to object. This new representation has a wide range of
applications. For instance, an image can be viewed as a set of prominent regions or points of interest, each with its individual feature vector [41]. A video is a sequence of video frames, each of which corresponds to a feature point [118]. Similarly, a text document contains a number of paragraphs from each of which an array of word occurrence frequencies are extracted [108]. A point set is also capable to represent a molecule in the study of drug activities [63] and to characterize voxelized CAD objects [91]. Moreover, in the applications over the database of points like clustering, it is useful to measure the proximity between different subsets of points [81], which are naturally examples of applications of the MPM. Generally, this new type of representation is important in a wide range of practices. The classification problem on the MPM data is to generate a classifier that can predict labels for the unseen examples. This research problem is called *Multiple-Instance Learning*, which is an active direction in the learning community [108,63].

In this study, we investigate the problem of similarity search over the point-set databases in two aspects:

(i) We propose meaningful distance functions which better characterize the similarity of two point sets. A specified number of points in one set are selected, each of which is matched with a distinct point in the other set. The proximity of the two sets is the sum of distances between all the paired points in the optimal matching. The computation of the new distance function is formulated into the classical assignment problem.

(ii) We propose lower bounding techniques to quickly discard majority of unqualified database items in the evaluation of nearest neighbor queries. Approximate matchings are derived efficiently for two given point sets and the lower bounds can be computed accordingly. We also propose to extract a small set of bounding rectangles for each point set as its coarse representation and define a bounding distance based on these rectangles. Furthermore, we design an early-stop mechanism to drop the expensive computations in the solver of the assignment problem as much as possible.
Our experimental results confirm the superiority of our proposals in terms of retrieval accuracy and efficiency.

The rest of the chapter is organized as follows. We discuss the technical background of the research problem in Section 5.2 and define the new distance and the similarity query in Section 5.3 and 5.4. The efficient query evaluation techniques are presented in Section 5.5. The experimental results are reported in Section 5.6. Finally, we conclude the chapter in Section 5.7.

5.2 Background

We first introduce notations of the multiple point model. In the MPM database $\mathcal{X}$, an object $X \in \mathcal{X}$ is a set of $|X|$ vectors in the $d$-dimensional space:

$$X = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{|X|}\},$$

where $|X|$ is the size of the set (i.e., the number of points in the set) and $\vec{x}_i \in \mathbb{R}^d$ and $x_{ij}$ is the $j$th component of the $i$th point in $X$. A vector $\vec{x}$ belongs to a set $X$, denoted as $\vec{x} \in X$. Consider $X$ and $X'$, there are totally $|X| \times |X'|$ pairs of points between every vector $\vec{x}_i \in X$ and $\vec{x}_j' \in X'$. The distance of a pair of points can be determined as the norm of their differences, $\|\vec{x}_i - \vec{x}_j\|$, just like in the single point model. Throughout the study, the Euclidean distance is adopted as the base distance metric of two points due to its effectiveness and popularity [124], that is, $\|\vec{x}_i - \vec{x}_j\| = \sqrt{\sum_{k=1}^{d} |x_{ik} - x_{jk}'|^2}$.

The directed maximal Hausdorff distance [53] is defined to measure the similarity of two sets of points:

$$d_{h,M}(X, X') = \max_{\vec{x} \in X} \min_{\vec{x}' \in X'} \|\vec{x} - \vec{x}'\|.$$
And its generalized version is the \( r \)-th ranked distance (\( r \) is a user-specified parameter) [131],

\[
d_{hr}(X, X') = r\text{th} \min_{\vec{x} \in X} \min_{\vec{x}' \in X'} \|\vec{x} - \vec{x}'\|.
\]

If \( r = 1 \), the distance \( d_{h1} \) is the \textit{minimal Hausdorff distance}, the smallest value of all the pairwise distances,

\[
d_{h1}(X, X') = \min_{\vec{x} \in X} \min_{\vec{x}' \in X'} \|\vec{x} - \vec{x}'\|.
\]

Generally, the directed Hausdorff distance are not symmetric, i.e., \( d_{hr}(X, X') \) is not necessarily equal to \( d_{hr}(X', X) \). The maximum of these two values can be adopted to be a symmetric distance of point sets. Basically, to compute the Hausdorff distance is to rank all pairwise distances between points in \( X \) and \( X' \) and choose one of them as the distance value. This family of distance functions have been shown to have high accuracy in the nearest neighbor classification, especially the distance \( d_{h1} \) [149], however, it is very sensitive to outliers [131]. To address this issue, the average of all the pairwise distances can be used as the similarity measurement of point sets [81],

\[
d_A(X, X') = \frac{1}{|X| \times |X'|} \sum_{\vec{x} \in X} \sum_{\vec{x}' \in X'} \|\vec{x} - \vec{x}'\|.
\]

Another metric \( d_C \) is defined based on the centers of the points of two sets,

\[
d_C(X, X') = \left\| \left( \frac{1}{|X|} \sum_{\vec{x} \in X} \vec{x} \right) - \left( \frac{1}{|X'|} \sum_{\vec{x}' \in X'} \vec{x}' \right) \right\|.
\]

The Ward’s metric is the distance \( d_C \) normalized with regard to the sizes of \( X \) and \( X' \),

\[
d_W(X, X') = \sqrt{ \frac{|X||X'|}{|X| + |X'|} d_C^2(X, X') }.
\]
The three measurements \((d_A, d_C, \text{ and } d_W)\) have been adopted in the bottom-up hierarchical clustering algorithm [84] over the database of points to measure the similarity between different subsets of points.

In recent proposals, points in two sets are paired according to different criteria. For example, the (fair-)surjection and the flow network between two sets are respectively introduced in [64] and [106]. The proximity of point sets is then defined based on the optimal pairings. However, these distance functions are computational prohibitive and the retrieval performances are mediocre [91,106]. To compare two point sets of the same size, a vector in one set is matched with a distinct vector in the other set and the sum of distances in the matched pairs is used as the similarity measurement of the point sets [91]. If two objects have different sizes, the smaller set is augmented with zero vectors so that the two augmented objects to be compared have the same number of points. This distance definition is referred to be the full matching distance, denoted as \(d_{FM}\), which requires all the points in the augmented objects are matched and taken into the computation of the similarity value. The Earth Mover’s Distance (EMD) [110], denoted by \(d_{EMD}\), is originally designed to compare two histograms [135] and can be extended as the metric for points sets where each point has the same weighting factor within a set.

### 5.3 The Matching Distance

In this section, we examine a toy example which motivates our study and introduce the new distance function and discuss its optimality in the proposed data model and present the way to compute the distance value in polynomial time.

#### 5.3.1 A Toy Example

Consider a toy example of 3 images as shown in Figs. 5.1.a - 5.1.c. Each image is segmented into a number of regions and each region is characterized by its color and shape features.
Figure 5.1: Toy examples.

For example, the first image, $I_1$, is represented as a set of 3 feature vectors, corresponding to the black circle, blue square and white background. Similarly, each of the images $I_2$ and $I_3$ is denoted as 5 points respectively. These feature vectors are plotted in Fig. 5.2, in which the two axes correspond to the color and shape dimensions.

Figure 5.2: Feature vectors of the segmented regions in Images $I_1$, $I_2$ and $I_3$.

If a user is interested in finding images having a black circle and a blue square, he may use the image $I_1$ as an example in the query. In this case, the $I_3$ is the desired result. The 2 of the 3 vectors of $I_1$, representing the circle and square in $I_1$, can well match with the corresponding 2 vectors of $I_3$, which also represent the circle and square in $I_3$. This is a partial matching between the point sets of $I_1$ and $I_3$, that is, only a subset of points in $I_1$ are paired with those in $I_3$. However, none of the distance functions discussed in Section
5.2 can well express the semantic to support partial matching. This motivates us to design
a new distance function for the MPM data to allow flexible matchings.

5.3.2 The Distance Function and Its Optimality

We introduce a synthetic data model of point sets. Based on this model, we design the new
distance function and prove its optimality. Consider a synthetic two-class MPM database,
vectors of the objects are sampled from one of the following $d$-dimensional probability dis-
tributions:

(i) The $m$ normal distributions $\mathcal{N}(\bar{\mu}_i, I)$, $(1 \leq i \leq m)$, each having the mean vector
$\bar{\mu}_i \in \mathbb{R}^d$ and identity covariance matrix $I$;

(ii) The uniform distribution $\mathcal{U}$ in the bounding box $[-\bar{a}, \bar{a}]$, in which $\bar{a}$ is a $d$-dimensional
vector and each of its components is a sufficiently large constant value $a$, i.e., $\bar{a} = \langle a, \ldots, a \rangle \in \mathbb{R}^d$.

A generated point set is considered positive if it has at least one vector sampled from one
of the $m$ normal distributions. Otherwise, it is negative. In other words, objects of the
positive class must have points sampled from each of the $m$ normal distributions respectively.
This is a generalized model of the MPM data [41, 134]. With regard to this model, the
distances discussed in Section 5.2 between two positive examples may not be reliable since
each example can have some number of points sampled from the uniform distribution $\mathcal{U}$,
which are considered as noises. In general, these distance functions are not very meaningful
to recognize similar objects in the above model.

Consider positive examples $X = \{\bar{x}_1, \ldots, \bar{x}_{|X|}\}$ and $X' = \{\bar{x}'_1, \ldots, \bar{x}'_{|X'|}\}$. As each of them
has points sampled from the $m$ normal distributions, let $\tau_1, \ldots, \tau_m$ be $m$ distinct subscripts
for $X$ such that, for $1 \leq i \leq m$, we have,

$$\bar{x}_{\tau_i} \sim \mathcal{N}(\bar{\mu}_i, I),$$

96
which indicates that the $\tau_i$-th vector of $X$ follows the normal distribution $\mathcal{N}(\vec{\mu}_i, I)$. Similarly, we have the subscripts $\tau_i'$ for $X'$ such that, for $1 \leq i \leq m$, $\vec{x}_{\tau_i'} \sim \mathcal{N}(\vec{\mu}_i, I)$. Given the two subscripts, each of the $m$ points $\vec{x}_{\tau_i}$ in $X$ is matched with $\vec{x}_{\tau_i'}$ in $X'$, as they two are sampled from the same normal distribution. The subscripts form a matching between the two sets, which has $m$ point pairs. Accordingly, the (dis)similarity value between $X$ and $X'$ is the sum of the squared distances of these $m$ pairs, that is,

$$\sum_{i=1}^{m} \| \vec{x}_{\tau_i} - \vec{x}_{\tau_i'} \|^2.$$ 

According to the maximum likelihood theory \cite{114}, a robust distance metric is to maximize the similarity probability for a given data distribution. As shown in the proof in Section 5.8.1, the above distance function is exactly the maximum likelihood estimator of the synthetic model. Hence, the defined distance is the optimal and robust ranking metric for the above model \cite{76}. It is true that two point sets with a small distance have a high possibility to belong to the same class.

In practice, for two point sets $X$ and $X'$, the subscripts $\tau_i$ and $\tau'_i$ are implicit and hidden. There are potentially a large number of matchings between points in $X$ and $X'$. Among all the matchings, the one that minimizes the matching cost is of particular interest and importance to us. The similarity measurement is then determined by the least-cost mapping of they two. Formally, the matching distance between $X$ and $X'$ with regard to an optimal matching of $m$ point pairs is

$$d_M^m(X, X') = \min_{\tau_i, \tau_i'} \sum_{i=1}^{m} \| \vec{x}_{\tau_i} - \vec{x}_{\tau_i'} \|^2,$$

where $m$ is called the matching cardinality, which must be no greater than $\min(|X|, |X'|)$; $\tau_i$ and $\tau_i'$ denote the $m$ distinct subscripts of the vectors of $X$ and $X'$ respectively.
Since the base distance is a metric, it is guaranteed that the distance $d_M^m$ is non-negative and symmetric. The distance of two point sets being zero does not indicate these two are the same, having the same set of vectors. The matching distance does not satisfy the triangle inequality property. Consider three point sets, $X = \{\langle 1 \rangle, \langle 2 \rangle \}$, $X' = \{\langle 1 \rangle, \langle 3 \rangle \}$, and $X'' = \{\langle 3 \rangle \}$, we have $d_M^1(X, X')$ equal to zero with $m = 1$ while $X \neq X'$. The other distance values are as follows, $d_M^1(X', X'') = 0$ and $d_M^1(X, X'') = 1$, and the triangle inequality property does not hold, since

$$d_M^1(X, X') + d_M^1(X', X'') < d_M^1(X, X'').$$

In case that the matching cardinality $m$ is 1, the distance $d_M^1$ is essentially equivalent to the minimal Hausdorff distance $d_{h1}$, the smallest value of all the pairwise distances between two point sets. Consider a given MPM object $X$, we can expand the set $X$ by adding $k$ constant vectors $\vec{c} \in \mathbb{R}^d$ and the augmented set is denoted as:

$$X^{k, \vec{c}} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}|X|, \underbrace{\vec{c}, \ldots, \vec{c}}_k\}.$$

The full matching distance $d_{FM}$ [91], discussed in Section 5.2, can be viewed as a special case of our distance function over augmented objects,

$$d_{FM}(X, X') = d_M^{\max(|X|, |X'|)}(X^{\max(0, |X'| - |X|), \vec{0}}, X'^{\max(0, |X| - |X'|), \vec{0}}).$$

As shown in the above equation, the point sets to be compared are augmented with zero vectors $\vec{0} \in \mathbb{R}^d$ so that the new sets have the same number of vectors, that is $\max(|X|, |X'|)$. The matching cardinality is set to be the size of the augmented objects. Thus, a point in one set is paired with a distinct point in the other set and the distances of all the pairs are added to the matching cost. This is a full matching and the distance $d_{FM}$ is sensitive to
noises. Consider two positive objects in the synthetic model, vectors in one set sampled from the uniform distribution are matched with some vectors in the other set and the distances of these point pairs can potentially distort the similarity value. To address this issue, our proposal is a generalized matching distance, supporting partial matchings, in which only a user-specified number of pairs are taken into computation of the similarity value.

Let us review the toy example discussed before. A user can set the matching cardinality to be 2 to express his intent since he is looking for two components (a circle and a square). As $m = 2$, the distance $d^2_M(I1, I3)$ is zero and smaller than $d^2_M(I1, I2)$, therefore $I3$ is more similar to $I1$ and returned to the user. When $m = 1$, the matching distance is small for two objects having one pair of similar components. In the example, the 3 images have zero distances to each other, i.e., $d^1_M(I1, I2) = d^1_M(I1, I3) = d^1_M(I2, I3) = 0$. If $m$ is set to be 3, the distance between $I1$ and $I3$ is smaller than that of $I1$ and $I2$. The distance between $I2$ and $I3$ is zero because the three regions (rectangle, square and triangle) of $I2$ can well match the corresponding regions in $I3$. This example has demonstrated the capability of our new distance function to support partial matchings.

### 5.3.3 The Evaluation of the Distance Function

We first define the virtual zero-distance vector, denoted as $\vec{0}^v$, which is a superficial vector and has zero distance to any vectors, i.e., for any $\vec{x} \in \mathbb{R}^d$, it is true that $\|\vec{x} - \vec{0}^v\| = 0$.

Two point sets to be compared are usually of different sizes. There is a user-specified parameter in our distance definition, which is the matching cardinality $m$. By introducing the virtual vector $\vec{0}^v$, we manage to transform the problem to compute the matching distance into the classical assignment problem [112], which is the agent-task assignment problem in its general form, that is, to minimize the overall cost to assign tasks to agents. Mathematically, the assignment problem is:
Problem 2  Given a square cost matrix $W$ of size $n \times n$, find a permutation $\tau$ of the $n$ integers $\{1, 2, \ldots, n\}$ to minimize the total cost,

$$\min_{\tau} \sum_{i=1}^{n} W(i, \tau_i).$$

Intuitively, the value $W(i, j)$ is the cost for the $i$th agent to do the $j$th job. The optimal $\tau_i$ indicates that the $\tau_i$-th task is assigned to the $i$th agent, and this minimizes the overall cost.

With the virtual vector $\vec{0}^v$, we have the following conclusion:

Theorem 5.1  Given two point sets $X$ and $X'$, and $m \leq \min(|X|, |X'|)$, the distance $d^m_M(X, X')$ is equal to the minimal cost of the assignment problem with the matrix $W$, that is the pairwise distance matrix between points of $X^{(|X'|-m),\vec{0}^v}$ and $X'^{(|X|-m),\vec{0}^v}$, i.e., for $1 \leq i, j \leq (|X| + |X'| - m),

$$W(i, j) = \begin{cases} \|\vec{x}_i - \vec{x}_j\|^2 & \text{if } 1 \leq i \leq |X|, 1 \leq j \leq |X'|, \\ 0 & \text{otherwise.} \end{cases}$$

Proof:  Omitted.

The objects $X^{(|X'|-m),\vec{0}^v}$ and $X'^{(|X|-m),\vec{0}^v}$ are of the same size, that is, $|X| + |X'| - m$. In general, the added $(|X'| - m)$ virtual vectors in $X^{(|X'|-m),\vec{0}^v}$ are designed to match with $(|X'| - m)$ vectors in $X'$ with zero cost, and similarly, the added vectors in $X'^{(|X|-m),\vec{0}^v}$ are paired with $(|X| - m)$ vectors in $X$. Consequently, the rest unmatched $m$ vectors in $X$ are mapped to the unmatched ones in $X'$ and the matching cardinality is exactly $m$. The sum of the distances of these $m$ point pairs, the matching distance between $X$ and $X'$, is the result of the assignment problem with the matrix $W$, because the padded virtual vectors contribute zero cost to the result.

As shown in Theorem 5.1, to compute the matching distance $d^m_M$ essentially requires the solver of the assignment problem. The Hungarian Algorithm is the polynomial time ($O(n^3)$)
algorithm for the linear assignment problem, in which \( n \) is the row (column) dimension of the square cost matrix \( W \). We have implemented the Hungarian algorithm that computes the least-cost assignment using vertex labeling [112] and leveraged the solver to compute the matching distance.

5.4 Similarity Query in Point-Set Databases

In the similarity search against a database of point sets \( \mathcal{X} \), the object of interest is denoted as \( Q = \{\vec{q}_1, \ldots, \vec{q}_{|Q|}\} \) (called the query point set). The \( k \)-nearest neighbor query is to find the \( k \) objects in the database closest in distance to the query object \( Q \). Below, we discuss two different ways to formulate the \( k \)-nearest neighbor search in the MPM data.

In the query formulation, a user needs to specify a parameter, the matching cardinality \( m \) (\( 1 \leq m \leq |Q| \)), besides the query point set \( Q \). In some applications, like image retrieval, a user can identify and eliminate some insignificant regions (feature vectors) from an image and use the rest feature vectors of the image as the query point set \( Q \), and the matching cardinality is the size of \( Q \), i.e., \( m = |Q| \). In some other applications, a user may use the domain knowledge or trial-and-error to determine a suitable matching cardinality with regard to the given point set \( Q \). In practice, it is likely that the size of an object \( X \) in the database is smaller than \( m \). In this case, no enough points of \( X \) can be selected to match with \( m \) points in \( Q \) in order to compute the \( d_M^m \) distance. To address this problem, we can add some vectors to \( X \) (\( |X| < m \)) so that the newly created object has \( m \) points to be paired with \( Q \).

In the first way, an object \( X \) is padded with virtual vectors \( \vec{0}^v \), and the derived object is \( X^{(m-|X|),\vec{0}^v} \). This is actually to ignore some of the selected \( m \) points of \( Q \) in the matching since these vectors are matched with \( \vec{0}^v \) at zero cost. It is true that, if \( m \geq |X| \),

\[
d_M^m(Q, X^{(m-|X|),\vec{0}^v}) = d_M^{|X|}(Q, X). \tag{Eq. 5.1}
\]
### Algorithm 5.1 Naive Evaluation of Nearest Neighbor Search

**Require:** a dataset $\mathcal{X}$, query $Q$ and parameter $k$, distance function $d_{M,1}^{m}$ (or $d_{M,2}^{m}$), the matching cardinality $m$.

**Ensure:** the top $k$ objects in $\mathcal{X}$ closest to $Q$ in distance.

1: Create a minimum priority queue.
2: for all $X$ in the database $\mathcal{X}$ do
3: Compute the cost matrix $W$ based on the given $Q$, $X$ and distance function.
4: Run the Hungarian solver on $W$ to compute the distance from $Q$ to $X$, denoted as $d$.
5: Push the entry $(d, X)$ into the queue.
6: end for
7: Return the top $k$ entries of the queue.

This approach may favor point sets of smaller sizes in the dataset, as they are likely to have small distances to the query set. To address this issue, instead of adding the virtual vector $\vec{0}^v$, an example $X$ can be augmented with some real vectors $\vec{c} \in \mathbb{R}^d$. The point set $X^{k,\vec{c}}$ contains all the points of $X$ and $k$ added vectors $\vec{c}$. Some of the selected $m$ points in $Q$ are then paired with $\vec{c}$. A suitable choice of $\vec{c}$ is the origin of the $d$-dimensional space, the zero vector $\vec{0} = (0, \ldots, 0) \in \mathbb{R}^d$. The generated point set is denoted as $X^{k,\vec{0}}$ and used in the matching. In summary, our defined similarity query is to identify $k$-nearest neighbors of $Q$ with matching cardinality $m$ ($1 \leq m \leq |Q|$) using one of the below distance functions:

\[
\begin{align*}
    d_{M,1}^{m}(Q, X) &= d_{M}^{m}(Q, X^{(\min(0,m-|X|)),\vec{0}^v}) = d_{M}^{m(|X|,m)}(Q, X), \\
    d_{M,2}^{m}(Q, X) &= d_{M}^{m}(Q, X^{(\min(0,m-|X|)),\vec{0})}. 
\end{align*}
\]

In the above similarity measurements, an object $X$ in the database is padded with some vectors (either the virtual vector $\vec{0}^v$ or the actual zero vector $\vec{0}$) if $X$ has fewer points than $m$, so that the size of the new point set $X^{(\min(0,m-|X|)),\vec{0}^v}$ (or $X^{(\min(0,m-|X|)),\vec{0})$) is at least no smaller than the matching cardinality of the query. Between the new object and the $Q$, $m$ pairs of points are selected in order to minimize the total sum of distances of the $m$ selected pairs. If $m$ is smaller than the sizes of $Q$ and $X$, then $d_{M,1}^{m}(Q, X)$ is equal to $d_{M,2}^{m}(Q, X)$. Otherwise, it is guaranteed that the distance $d_{M,1}^{m}(Q, X)$ is no greater than $d_{M,2}^{m}(Q, X)$. 

102
In terms of computational efficiency, a straightforward implementation of the above distance functions requires one full run of the Hungarian algorithm per distance computation. To retrieve \( k \) nearest neighbors of the query object \( Q \) from the database \( X \), the naive approach is to examine all the point sets in \( X \), as shown in Algorithm 5.1. The minimum priority queue is the queue to store the elements in ascending order of distances, which is used to keep the current top \( k \) candidates of \( Q \) during query evaluation. This is very computational expensive. In the next section, we investigate different methods to substantially reduce the expensive cost of query processing.

### 5.5 Efficient Query Evaluation

In this section, we introduce three lower bounding approximation techniques and discuss the multi-step query processing algorithm. We propose the early-stop mechanism to accelerate the evaluation of nearest neighbor queries.

#### 5.5.1 The First Lower Bounding

As the distance function \( d_{M,1}^m(Q, X) \) is equal to the matching distance between \( Q \) and \( X \) with the matching cardinality \( \min(|X|, m) \), as in Eq. 5.1. Therefore, the proposed distance metrics \( (d_{M,1}^m \text{ and } d_{M,2}^m) \) are essentially computed based on the matching distance \( d_{M}^m \) between the point sets which consist of only real vectors. In the below discussions, we focus on the lower bounding function for the distance \( d_{M}^m(X, X') \) with the matching cardinality \( m \leq \min(|X|, |X'|) \) and \( X, X' \) are sets of real vectors.

The matching distance \( d_{M}^m \) with cardinality \( m \) is to select \( m \) distinct points from \( X \) and \( m \) distinct points from \( X' \) and create a one-to-one mapping between the two sets of selected points so that the sum of the distances of these \( m \) pairs of points in the mapping is minimized. This requires the mapping to be feasible, that is, the partner of each of the selected \( m \) points in \( X \) comes from \( X' \) and should be different from each other and vice versa. We can define
Algorithm 5.2 Lower Bounding By Nearest Neighbors Matching

Require: point sets $X, X'$ and cardinality $m$.
Ensure: the lower bounding value.

1: Create a minimum priority queue.
2: for all $\vec{x} \in X$ do
   3: Find the nearest neighbor of $\vec{x}$ in $X$, that is, $\vec{y} = \arg\min_{\vec{x}' \in X'} \|\vec{x} - \vec{x}'\|^2$.
   4: Push $(d, \vec{x})$ into the queue where $d = \|\vec{x} - \vec{y}\|^2$.
5: end for
6: Return the sum of the distances of the top $m$ entries in the queue.

A matching by allowing each vector $\vec{x} \in X$ to pair with its nearest neighbor in $X'$. Note that, this matching is not necessarily feasible as two different points in $X$ may have the same vector in $X'$ as their nearest neighbor. We can select the $m$ pairs of smallest distances from this matching to create a new matching of the cardinality $m$. The sum of these $m$ distances is guaranteed to underestimate the value $d_M^m(X, X')$. The whole procedure to compute the lower bound is shown in Algorithm 5.2, and the return value is denoted as $d_{lw1}^m(X, X')$. The time complexity of the algorithm is $O(|X| \times |X'|)$, which is one order of magnitude smaller than that of the Hungarian algorithm. Note that, $d_{lw1}^m(X, X')$ is not necessarily equal to $d_{lw1}^m(X', X)$, and the maximum of these two values is a tighter lower bound of the matching distance,

$$\max(d_{lw1}^m(X, X'), d_{lw1}^m(X', X)) \leq d_M^m(X, X').$$

For example, $X = \{(1), (2), (5)\}$ and $X' = \{(-1), (1), (1)\}$, then we have $d_M^3(X, X') = 21$, and $d_{lw1}^3(X, X') = 17$, while $d_{lw1}^3(X', X) = 4$.

### 5.5.2 The Second Lower Bounding

To compute the distance $d_M^m(X, X')$, according to $m$ and the sizes of the two objects $|X|$ and $|X'|$, the cost matrix $W$ is created accordingly. The input of the second lower bounding method (Algorithm 5.3) is this square matrix $W$ with $n$ rows and $n$ columns. The algorithm
takes the matrix and tries to create a feasible matching between the rows and the columns of the matrix. First, it scans the entire matrix to locate the smallest entry, suppose this cell is the intersection of the \( i \)th row and \( j \)th column, i.e., \( W(i, j) \). The cell is selected and its value is added to the bound. This indicates that the \( i \)th row is matched with the \( j \)th column. Then the values of all the entries in the \( i \)th row and \( j \)th column are set to be the positive infinity, which makes the \( i \)th row and the \( j \)th column become unavailable for the subsequent selections. This process iterates until all the rows and columns are matched. It is easy to see that the matching generated eventually is a one-to-one mapping and feasible, and therefore, the returned value is an upper bound of the result of the assignment problem, which is the minimum among all the feasible matchings. We have proved that Algorithm 5.3 is the 2-approximation algorithm for the assignment problem (the proof is available in Section 5.8.2), which satisfies,

\[
\frac{1}{2} \times \text{bound} \leq \min_{\tau} \sum_{i=1}^{n} W(i, \tau_i) = d_{M}^m(X, X') \leq \text{bound},
\]

where \( \text{bound} \) is the value returned by Algorithm 5.3. Therefore, we define a new lower bounding \( d_{lw2}^m(X, X') \) to be the half of the bounding value returned by Algorithm 5.3 for the distance \( d_{M}^m(X, X') \). This algorithm can be implemented efficiently with the time complexity \( O(n^2 \log n) \) by first sorting the entries on each row individually and then the selection of the smallest cells can be done quickly using the sorted results instead of scanning the whole matrix for \( n \) times.

### 5.5.3 Minimum Bounding Rectangle Approximation

The time complexity of the solver of the matching distances and the related lower bounding algorithms is proportional to the sizes of the input point sets. We consider to use approximate representations of the point sets to create an approximate matching to compute a lower bound efficiently. Specifically, an object \( X \) has a number of points. For each \( X \), its points can be
Algorithm 5.3 Lower Bounding By Nearest Feasible Neighbor Matching

Require: a square cost matrix $W$ of size $n \times n$.

Ensure: the bounding value.

1: Set $bound = 0$.
2: for all $i$ such that $1 \leq i \leq n$ do
3: Get the smallest entry $W(i, j)$ in the matrix $W$.
4: Add the value $W(i, j)$ to $bound$.
5: Set all the values in the $i$th row and $j$th column of the matrix $W$ into positive infinity.
6: end for
7: Return the value $bound$.

arranged into a small number of subgroups. The number of the subgroups is much smaller than the size of a point set and we can compute the distance of a greedy matching between subgroups of $X$ and $X'$ quickly.

Suppose the $|X|$ points of $X$ are divided into $m_X$ subgroups $S^X_i$ ($1 \leq i \leq m_X$), that is, $\bigcup_{i=1}^{m_X} S^X_i = X$ and none of two subgroups are overlapped, for $1 \leq i \neq j \leq m_X$, $S^X_i \cap S^X_j = \emptyset$. The size of the subset $S^X_i$ is the number of points it has, denoted as $|S^X_i|$. Each subgroup of points can be represented using its Minimum Bounding Rectangle (MBR) [25], that is the compact box along the coordinate axes to contain all the points in $S^X_i$. The set of MBRs of the example $X$ is denoted as $MBR^X$ and its $i$th component, the bounding box of the $i$th subset $S^X_i$ is characterized as,

$$MBR^X_i = (\bar{l}^X_i, \bar{u}^X_i, |S^X_i|),$$

where $\bar{l}^X_i$ and $\bar{u}^X_i$ respectively represent the lower and upper bounds of the projections of points in $S^X_i$ along the coordinate axis, that is, for $1 \leq i \leq m_X, 1 \leq j \leq d$, 

$$l^X_{ij} = \min(\{x_j \mid \bar{x} \in S^X_i\}),$$

$$u^X_{ij} = \max(\{x_j \mid \bar{x} \in S^X_i\}).$$
There are $|S^X_i|$ vectors in the corresponding bounding rectangle. The minimum distance of the two MBRs, $MBR^X_i$ and $MBR^X_j'$, is the smallest distance between any point in $MBR^X_i$ and any point in $MBR^X_j'$, that is,

$$\text{mindist}(MBR^X_i, MBR^X_j') = \sqrt{\sum_{k=1}^d (\text{mindist}_k(MBR^X_i, MBR^X_j'))^2},$$

where, for $1 \leq k \leq d$,

$$\text{mindist}_k(MBR^X_i, MBR^X_j') = \begin{cases} 
\|l^X_{ik} - u^X_{jk}'\| & \text{if } l^X_{ik} > u^X_{jk}', \\
\|l^X_{jk}' - u^X_{ik}\| & \text{if } u^X_{ik} < l^X_{jk}', \\
0 & \text{otherwise.}
\end{cases}$$

We can compute the subgroup partitioning and build the MBR representations $MBR^X$ for each object $X$ in the MPM database offline, which will be discussed in detail at the end of this subsection. To compute the distance function $d^m_{M,2}$, an object to be examined in the database may need to be padded with some real vectors. The MBRs of this newly padded set can be generated online. Suppose in the query time, the point set $X$ is enlarged with $k$ vectors $\vec{0}$. The set of MBRs of the new object $X^{k,\vec{0}}$ has one more rectangle than $X$,

$$MBR^{X^{k,\vec{0}}} = \{MBR^X, (\vec{0}, \vec{0}, k)\},$$

where the new bounding rectangle solely consists of $k$ vectors $\vec{0}$. Consequently, it is feasible to use the MBR representations of point sets in the query time to compute the lower bound of the matching distance, as described below.

Consider to compute the exact distance $d^m_{M\tau}(X, X')$ with $m \leq \min(|X|, |X'|)$ and $X$ and $X'$ consist of only real vectors. Suppose $m$ points are selected from the set $X$, whose subscripts are denoted as $\tau_i$ ($1 \leq i \leq m$). The set of the selected $m$ vectors are $X_\tau = \{\vec{x}_{\tau_i} | 1 \leq i \leq m\}$. The points of $X_\tau$ which also belong to the $i$th subgroup $S^X_i$ are $X_{\tau,i} = S^X_i \cap X_\tau$, 

107
and they are inside the bounding rectangle,

\[ MBR^{X_{\tau,i}} = (\vec{l}^X_{\tau,i}, \vec{u}^X_{\tau,i}, |X_{\tau,i}|), \]

having the same lower and upper bounds as the \( MBR^X \). The approximate matching distance between \( X_{\tau,i} \) and \( X' \), denoted as \( d_{MA}(X_{\tau,i}, X') \), is the smallest cost to match points of \( X_{\tau,i} \) with those in \( X' \) based on their MBR approximations \( MBR^{X_{\tau,i}} \) and \( MBR^{X'} \). The minimum distances of the bounding rectangles of points are used to compute the approximate matching distance instead of the exact base distances between the points. Specifically, the distance \( d_{MA} \) is the optimal cost of the assignment problem with the matrix \( W \), where for \( 1 \leq k, j \leq \max(|X_{\tau,i}|, |X'|) \),

\[
W(k, j) = \begin{cases} 
\left( \min\text{dist}(MBR^{X_{\tau,i}}, MBR^{X'}) \right)^2 & \text{if } 1 \leq k \leq |X_{\tau,i}|, 1 \leq j \leq |X'|, \\
0 & \text{otherwise.}
\end{cases}
\]

in which \( \gamma'_j \) denotes the index of the subgroup that the vector \( \vec{x}'_j \in X' \) belongs to, i.e., \( \vec{x}'_j \in S_{\gamma'_j}^{X'} \). In case that \( X_{\tau,i} \) is an empty set, the value \( d_{MA}(X_{\tau,i}, X') \) is set to zero. It can be proved that, for \( 1 \leq i \leq m_X \),

\[
d_{MA}(X_{\tau,i}, X') \leq d_{M}^{X_{\tau,i}}(X_{\tau,i}, X').
\]

The exact matching distance of \( X_{\tau,i} \) and \( X' \) is guaranteed to be no smaller than their approximate matching distance. We can compute the approximate distances between each \( X_{\tau,i} \) and \( X' \) and sum them together. The sum of all these derived distance values is a function of the selection \( \tau \) of \( m \) vectors from the set \( X \). In particular, we are interested in the selection that minimizes the sum of the approximate distance values,

\[
d_{lw3}(X, X') = \min_{\tau} \sum_{i=1}^{m_X} d_{MA}(X_{\tau,i}, X').
\]
Algorithm 5.4 Lower Bounding Based on the MBRs

Require: $MBR^X$ of $X$, $MBR^{X'}$ of $X'$, and $m$.
Ensure: the lower bounding value.

1: Initialize arrays $s_1$ and $s_2$,
   
   for $1 \leq i \leq m_X$, $s_1(i) = |S^X_i|$.
   for $1 \leq i \leq m_{X'}$, $s_2(i) = |S^{X'}_i|$.

Initialize matrix $W_{mbr}$, for $1 \leq i \leq m_X$, $1 \leq j \leq m_{X'}$,

$$W_{mbr}(i, j) = \left( \min_{i,j} \text{dist}(MBR_i^X, MBR_j^{X'}) \right)^2.$$ 

2: Set $bound = 0$ and $m' = 0$.
3: while $m' < m$ do
4:   Get the smallest entry $W_{mbr}(i, j)$ in the matrix $W_{mbr}$.
5:   Let $p = \min(m - m', s_1(i), s_2(j))$, and update values,
   
   $bound = bound + W_{mbr}(i, j) \times p.$
   $s_1(i) = s_1(i) - p.$
   $m' = m' + p.$
   $W_{mbr}(i, j) = +\infty.$
6: end while
7: Return the value $bound$.

This lower bound value $d_{lw3}^m$ is computed based on the bounding rectangles and guarantees to be no greater than the exact matching distance $d_M^m(X, X')$, because

$$d_{lw3}^m(X, X') = \min_{\tau} \sum_{i=1}^{m_X} d_{MA}(X_{\tau,i}, X')$$

$$\leq \min_{\tau} \sum_{i=1}^{m_X} d_M^{X_{\tau,i}}(X_{\tau,i}, X')$$

$$\leq d_M^m(X, X').$$

Algorithm 5.4 is the solver to compute the value $d_{lw3}^m(X, X')$. At the start, the algorithm computes the minimum distances between each pair of bounding rectangles of $X$ and $X'$, and the values are stored in the matrix $W_{mbr}$ in Line 1 of the algorithm. The variable $m'$ (in
Line 2) keeps track of the number of points already selected from the set $X$ at the current step. Initially, no points are selected. In every subsequent iteration, the algorithm identifies the cell of the smallest distance value $W_{mbr}(i, j)$ from the matrix $W_{mbr}$. This is the current smallest cost to match one point in $X_{\tau,i}$ with those in the bounding rectangle $MBR_j^{X'}$. Hence, as many points from $X_{\tau,i}$ as possible are selected to match with the points in the $j$-th MBR of $X'$ at the cost of $W_{mbr}(i, j)$ each, as in Line 5 of the algorithm. The bound variable is updated for the newly selected points in $X_{\tau,i}$ and the value of the cell $W_{mbr}(i, j)$ is set to infinity, that prohibits this cell from being selected again in the subsequent iterations.

The correctness of the lower bound function $d_{lw}^m$ and Algorithm 5.4 is discussed in detail in Section 5.8.3. The time complexity of the algorithm is $O(m_Xm_{X'} \log m_{X'})$, as it can be implemented with the same philosophy as Algorithm 5.4, by first sorting entries of the matrix and computing the lower bound based on the sorted values.

In this proposal, we use the hierarchical clustering algorithm based on the Ward’s criterion [81] to compute the subgroup partitioning $S_i^X$ for each point set $X$ in the database. Initially, each point is a cluster by itself, and in the subsequent steps, two intermediate clusters that are closest in the Ward’s metric $d_W$ are merged into a bigger cluster. This process carries on until the desired number of clusters has been generated. Each derived cluster of a point set $X$ constitutes a subgroup $S_i^X$. We use a simple heuristic to determine the number of clusters of a point set $X$. If $X$ has fewer than 3 points, then each of its points forms a subgroup. If the size of $X$ is between 4 and 10, the points are divided into 3 subgroups. Otherwise, they are partitioned into 5 subsets.

### 5.5.4 Multi-Step Nearest Neighbor Search

Different lower bounding functions described in the previous sections can be utilized to accelerate the query processing. In Algorithm 5.5, the function $d_{lw}^m$ is adopted. During the query processing, the MBR representation of $Q$ is created in the same fashion as $MBR^X$ of
Algorithm 5.5 Multi-Step Nearest Neighbor Search

**Require:** a dataset \( \mathcal{X} \), \( MBR^X \) of each \( X \in \mathcal{X} \)

**Require:** query \( Q \) and parameter \( k \), distance function \( d_{M,1}^m \) (or \( d_{M,2}^m \)), the matching cardinality \( m \).

**Ensure:** the top \( k \) objects in \( \mathcal{X} \) closest to \( Q \) in distance.

1: For \( Q \), create the subgroup partitioning \( S_i^Q \) and \( MBR^Q \).
2: Create a minimum priority queue.
3: for all \( X \) in the database \( \mathcal{X} \) do
4: Run Algorithm 5.4 for \( Q \) and \( X \) and the returned value is denoted as \( d_A \). Push \((d_A, X)\) into the queue.
5: end for
6: while TRUE do
7: Pop up the top entry of the queue, denoted as \((d, X)\).
8: if the distance \( d \) is the lower bounding value then
9: Run Hungarian solver to compute the exact matching distance \( d_E \). Push \((d_E, X)\) into the queue.
10: else
11: Add \((d, X)\) to the result list. If \( k \) objects have been found, break the while loop and return the result.
12: end if
13: end while

\( X \in \mathcal{X} \). All the objects in the database are examined with \( Q \) in the approximate matching distance \( d_{lw3}^m \). All these distance values are pushed back to a minimum priority queue. Entries in the queue are retrieved in order and the exact matching distances are computed until top \( k \) nearest objects have been found. In this way, unnecessary exact distance computations are avoided.

This algorithm follows the framework of the optimal multi-step \( k \)-nearest neighbor search [115]. Similarly, we can adopt other lower bounding functions to speed up the query evaluation. We can further combine the three functions introduced in the study, by first using \( d_{lw3}^m \) followed by \( d_{lw1}^m \) followed by \( d_{lw2}^m \).

\(^1\)According to the query semantic, \( Q \) and \( X \) may need to be padded with some vectors \( \vec{0} \) and the new objects are then used in the computation of Algorithm 5.4.
5.5.5 Early-Stop Mechanism

The three lower bounding approximation techniques aim to reduce the number of times the Hungarian solver need to run for a nearest neighbor query. Instead, we make use of a property of the Hungarian algorithm, as discussed below, to design an early-stop mechanism to reduce the cost in the execution of the solver during the query evaluation.

As described in Section 5.3.3, the Hungarian algorithm is to solve the agent-task assignment problem. Given the same number of agents and tasks, it computes the optimal one-to-one full matching between agents and tasks, that is, an agent is assigned to do a distinct task and the overall cost is minimized. The solver starts with an initial partial matching and gradually grows it into a complete one after many iterations of matching refinement. It is guaranteed that the cost of the matching in the intermediate steps increases steadily until the algorithm finds a complete matching [112]. This property can help us to safely skip unnecessary computations in running the solver.

During the query evaluation, the minimum priority queue keeps the current top \( k \) candidates of the query object \( Q \). Let \( d_k \) denote the distance between the current \( k \)-th nearest neighbor in the queue and \( Q \). If the priority queue has fewer than \( k \) items, then the value \( d_k \) is set to be infinite. When we execute the solver to compare a database object and the query object \( Q \), the solver runs for many rounds and the cost of the intermediate matching increases iteration by iteration. As soon as the cost of the intermediate matching exceeds \( d_k \), we can abort the execution of the Hungarian algorithm at this step and there is no need to carry on the computations further. This is because it is already certain that the minimum cost of a complete matching between this database object and \( Q \) is greater than the current top \( k \) objects in the queue, and hence the database object is surely not included in the \( k \) nearest neighbors of \( Q \).

The early-stop mechanism can be integrated with the lower-bounding methods. We only need to modify Step 9 of Algorithm 5.5. The solver takes the value \( d_k \) as the input besides
the cost matrix. If the solver runs for completion with the exact matching distance, then this value is pushed with the database object into the queue as usual. Otherwise, the computation of the solver is aborted in the middle, and no more actions need to be performed for this database object.

5.6 Experimental Results

5.6.1 Settings

We performed experiments on several datasets. In the Musk-1 dataset [100], a set of points describe the low-energy conformations of the molecule. The COREL dataset has 20 distinct classes of 100 images each [41]. An image is characterized as a set of segments, each of which is denoted by a 9-dimensional feature vector. The third dataset consists of 1543 natural scene images that belong to 5 different categories [1]. In the fourth and fifth datasets [10], an object respectively corresponds to atoms and bonds of a molecule in the mutagenesis research. The detailed information of these datasets is listed in Table 5.1. The columns respectively show the size, the dimensionality, the number of classes, the average number of points per object in each dataset used in the experiments.

<table>
<thead>
<tr>
<th>dataset</th>
<th>size</th>
<th>dim</th>
<th>classes</th>
<th>points per object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Musk-1</td>
<td>92</td>
<td>166</td>
<td>2</td>
<td>5.174</td>
</tr>
<tr>
<td>COREL Images</td>
<td>2000</td>
<td>9</td>
<td>20</td>
<td>3.974</td>
</tr>
<tr>
<td>NJU Images</td>
<td>1543</td>
<td>15</td>
<td>5</td>
<td>9.000</td>
</tr>
<tr>
<td>Mutagenesis Atom</td>
<td>188</td>
<td>10</td>
<td>2</td>
<td>8.606</td>
</tr>
<tr>
<td>Mutagenesis Bond</td>
<td>188</td>
<td>10</td>
<td>2</td>
<td>21.250</td>
</tr>
<tr>
<td>Trx-fold Protein</td>
<td>193</td>
<td>8</td>
<td>2</td>
<td>137.881</td>
</tr>
</tbody>
</table>

Table 5.1: Datasets used in the experiments.

We evaluated the retrieval accuracy of $k$-nearest neighbor search using our proposed matching distances compared with the other functions discussed in Section 5.2. We also
measured the computational efficiency of our different query evaluation schemes. The implementations of the Hungarian algorithm and the query processing techniques were coded in C++ and the MATLAB wrapper was also developed to measure the retrieval accuracy of different distance functions. All the experiments were performed on the computer with Core 2 Duo CPU E8500 3.16GHz and 4GB RAM running Windows Vista.

5.6.2 Retrieval Accuracy of Distance Metrics

Each of the objects in a dataset was used as the query point set and $k$-nearest neighbor queries were executed against the remaining objects in the dataset with different parameters of $k$. The retrieval accuracy of a $k$-nearest neighbor search is the percentage of the $k$ objects in the query result which have the same class label as the query object. This evaluation metric is also called the precision. The average accuracies of the queries were collected for each different distance metric and each examined value of $k$. In our defined matching distances ($d_{M,1}^m$ and $d_{M,2}^m$), the matching cardinality $m$ is meaningful, as it denotes how many points are selected from the query object and matched with those in the dataset. In general, a user is capable to tune the matching cardinality $m$ for a query. In the proposal, we used the automatic means to examine different parameters of $m$ for each query point set. The third quartile of retrieval accuracy of a query object among different $m$’s was collected and the results were then averaged over all the queries for our distance functions.

Our distance functions were compared with the other seven metrics discussed in Section 5.2. The average retrieval accuracy of the distances are summarized in Table 5.2. The performances are also plotted with regarding to different $k$’s in the $k$-nearest neighbor search, as shown in Figs. 5.3.a - 5.3.f. In the figures, only 7 methods are selected in order to keep the plots readable. The $y$-axis is the average accuracy, the indicator of the retrieval quality of different distance functions. The better the distance function, the higher its accuracy, the higher its curve. As shown in the experiments, our matching distances are much better than
5.3.a: Musk-1
5.3.b: COREL Images
5.3.c: NJU Images
5.3.d: Mutagenesis Atom
5.3.e: Mutagenesis Bond
5.3.f: Trx-fold Protein

Figure 5.3: Average retrieval accuracy of different distance metrics.
<table>
<thead>
<tr>
<th>dataset</th>
<th>$d_{h_1}$</th>
<th>$d_{h_M}$</th>
<th>$d_A$</th>
<th>$d_C$</th>
<th>$d_W$</th>
<th>$d_{FM}$</th>
<th>$d_{EMD}$</th>
<th>$d_{M,1}^m$</th>
<th>$d_{M,2}^m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Musk-1</td>
<td>75.2</td>
<td>73.3</td>
<td>72.2</td>
<td>69.6</td>
<td>71.5</td>
<td>67.6</td>
<td>72.2</td>
<td>78.1</td>
<td>78.0</td>
</tr>
<tr>
<td>COREL Images</td>
<td>37.8</td>
<td>48.1</td>
<td>23.0</td>
<td>39.6</td>
<td>37.5</td>
<td>52.0</td>
<td>55.9</td>
<td>49.2</td>
<td>60.5</td>
</tr>
<tr>
<td>NJU Images</td>
<td>48.9</td>
<td>50.0</td>
<td>43.9</td>
<td>50.1</td>
<td>50.1</td>
<td>50.6</td>
<td>50.6</td>
<td>59.3</td>
<td>59.3</td>
</tr>
<tr>
<td>Muta Atom</td>
<td>66.5</td>
<td>72.1</td>
<td>66.8</td>
<td>71.3</td>
<td>71.3</td>
<td>72.5</td>
<td>72.8</td>
<td>82.5</td>
<td>83.5</td>
</tr>
<tr>
<td>Muta Bond</td>
<td>67.7</td>
<td>72.0</td>
<td>68.1</td>
<td>77.1</td>
<td>76.9</td>
<td>76.8</td>
<td>78.7</td>
<td>83.0</td>
<td>84.5</td>
</tr>
<tr>
<td>Trx-fold Protein</td>
<td>75.4</td>
<td>77.8</td>
<td>67.3</td>
<td>78.3</td>
<td>78.7</td>
<td>81.1</td>
<td>79.3</td>
<td>88.2</td>
<td>90.8</td>
</tr>
<tr>
<td>Musk-1</td>
<td>60.5</td>
<td>55.8</td>
<td>56.7</td>
<td>55.4</td>
<td>55.6</td>
<td>57.2</td>
<td>56.6</td>
<td>65.5</td>
<td>61.8</td>
</tr>
<tr>
<td>COREL Images</td>
<td>31.4</td>
<td>36.6</td>
<td>20.0</td>
<td>32.5</td>
<td>30.5</td>
<td>39.5</td>
<td>44.2</td>
<td>39.1</td>
<td>48.2</td>
</tr>
<tr>
<td>NJU Images</td>
<td>45.2</td>
<td>43.7</td>
<td>40.4</td>
<td>44.5</td>
<td>44.5</td>
<td>46.2</td>
<td>46.2</td>
<td>51.4</td>
<td>51.4</td>
</tr>
<tr>
<td>Muta Atom</td>
<td>65.6</td>
<td>64.8</td>
<td>66.6</td>
<td>67.3</td>
<td>67.3</td>
<td>65.0</td>
<td>66.0</td>
<td>73.7</td>
<td>75.0</td>
</tr>
<tr>
<td>Muta Bond</td>
<td>65.9</td>
<td>65.6</td>
<td>66.9</td>
<td>69.8</td>
<td>69.1</td>
<td>72.9</td>
<td>69.8</td>
<td>75.8</td>
<td>78.8</td>
</tr>
<tr>
<td>Trx-fold Protein</td>
<td>78.3</td>
<td>78.2</td>
<td>71.3</td>
<td>78.2</td>
<td>78.4</td>
<td>80.0</td>
<td>79.8</td>
<td>84.1</td>
<td>88.9</td>
</tr>
</tbody>
</table>

Table 5.2: Average retrieval accuracy of different distance metrics.

the others. They are generally among the top two having higher query precision in different datasets and under different values of $k$. In particular, the distance $d_{M,2}^m$ is almost always the best over different $k$’s in the five datasets except the molecule dataset, Musk-1. It is evident that more objects relevant to the query object can be retrieved using our metrics.

### 5.6.3 Efficiency of Query Processing

For a dataset, 1000 pairs of query objects $Q$ and matching cardinalities $m$ were randomly generated. Nearest neighbor queries under the matching distance $d_{M,1}^m$ (or $d_{M,2}^m$) were executed for every pair of $Q$ and $m$ with different values of $k$ to study the computational efficiency of the query processing methods. We measured the average elapsed time per query for each of our techniques compared with the naive scan.

We reported the performances using the distance $d_{M,2}^m$, as the cost to evaluate queries with the two proposed distance functions were quite close. Figures 5.4.a - 5.4.f respectively plot the average elapsed time of different methods over different $k$’s in the datasets. The baseline is the naive evaluation method. In the Musk-1 dataset, the naive algorithm used
Figure 5.4: Efficiency of different query evaluation techniques.
around 0.432 second on average to scan the entire database per query in which 0.079 second was to compute pairwise distance matrixes between a query object and all the database objects, and the Hungarian solver took 0.353 second to compute the matching between the point sets. Our lower bounding functions used in the quick-and-dirty search only require to execute the costly Hungarian algorithm for a small number of database objects. The early-stop mechanism reduces the number of iterations of the solver during the query processing. Therefore, the elapsed time of these techniques is much smaller compared to the naive scan and their costs in query evaluation generally grow slowly with regarding to the $k$. The combined technique uses both the three lower bounding approximations and the early-stop mechanism, which is most effective in all datasets under different $k$’s. It consistently outperforms all other methods by a large margin in terms of efficiency.

5.7 Summary

We study the problem of similarity search over the database of point sets. We design meaningful distance metrics to better measure the similarity between point sets. For two given point sets, a specified number of points in one set are selected and each of them is matched with a distinct point in the other set. The distance of these two sets is defined to be the sum of distances of all the paired points between the two sets in the optimal matching. We also propose different lower bounding functions and the early-stop mechanism to accelerate the expensive evaluation of nearest neighbor queries over a database of point sets. The experimental results confirm the effectiveness and efficiency of our proposals.
5.8 Proofs

5.8.1 Maximum Likelihood Approach

The maximum likelihood theory can be leveraged to relate a distance function to a data distribution. The problem of finding a suitable metric for a data model is the maximization of the similarity probability.

Consider the synthetic model described in Section 5.2, two objects $X$ and $X'$ are considered similar if there are subscripts $\tau_i$ and $\tau'_i$ \(1 \leq i \leq m\),

\[
\vec{x}_{\tau_i} \sim \mathcal{N}(\vec{\mu}_i, I) \text{ and } \vec{x}'_{\tau'_i} \sim \mathcal{N}(\vec{\mu}_i, I).
\]

Then, we can define

\[
\vec{x}_{\tau_i} = \vec{x}'_{\tau'_i} + \vec{n}_i,
\]

where $\vec{n}_i$ is the vector of differences between two points sampled from the $i$th normal distribution. Hence, the differences $\vec{n}_i$ follow the zero-mean normal distribution,

\[
\vec{n}_i \sim \mathcal{N}(\vec{0}, 2I).
\]

In this context, we adopt the same philosophy as [114] to define the similarity probability between objects $X$ and $X'$,

\[
P(X, X') = \prod_{i=1}^{m} \exp \left( -\rho_i(\vec{x}_{\tau_i}, \vec{x}'_{\tau'_i}) \right), \quad (\text{Eq. 5.2})
\]

where the function $\rho_i$ is the negative logarithm of the similarity probability between the two vectors $\vec{x}_{\tau_i}$ and $\vec{x}'_{\tau'_i}$. To maximize the similarity probability, we need to minimize the
expression by taking the logarithm of Eq. 5.2:

$$\sum_{i=1}^{m} \rho_i(\vec{x}_{\tau_i}, \vec{x}'_{\tau'_i}).$$

The function $\rho_i$ depends on the difference between the two points, $\vec{n}_i$, that follows the normal distribution in our model. The similarity probability of the two vectors $P(\vec{x}_{\tau_i}, \vec{x}'_{\tau'_i})$ is proportional to the function $\exp \left( -\|\vec{x}_{\tau_i} - \vec{x}'_{\tau'_i}\|^2 \right)$. We have the below expression by taking out the constant factor,

$$\rho_i(\vec{x}_{\tau_i}, \vec{x}'_{\tau'_i}) = \|\vec{x}_{\tau_i} - \vec{x}'_{\tau'_i}\|^2.$$

Therefore, to minimize the sum of square deviations of the $m$ selected pairs of points,

$$\sum_{i=1}^{m} \|\vec{x}_{\tau_i} - \vec{x}'_{\tau'_i}\|^2,$$

is essentially the maximum likelihood estimator for our defined model. The smaller the sum of $m$ distances, the more similar the two objects.

### 5.8.2 Approximation Ratio of Algorithm 5.3

Algorithm 5.3 is the 2-approximation algorithm of the assignment problem, i.e., the value returned by Algorithm 5.3 is no greater than twice the optimal value of the assignment problem, $d_{\text{opt}}^m(X, X')$.

The assignment problem is the minimum weighted matching in the bipartite graph of which the edges are associated with costs in the input matrix $W$ of Algorithm 5.3. Suppose the optimal matching $M^*$ has the smallest cost value $W(M^*)$, and the corresponding optimal subscripts are denoted as $\tau^*_i$ and $\tau'^*_i$, that is, the vector $\vec{x}_{\tau^*_i}$ in $X$ is matched with $\vec{x}'_{\tau'^*_i}$ in $X'$, which corresponds to an edge in the graph. In each iteration of Algorithm 5.3, the smallest
feasible entry $W(e)$ is added to the matching $M$ where $e$ is an edge. This at most will delete two links $e_1$ and $e_2$ in the $M^*$. We have,

$$W(e) \leq W(e_1) \text{ and } W(e) \leq W(e_2).$$

All the removed edges $e_1$ and $e_2$ in each step are a superset of the edges in $M^*$. Therefore, we have,

$$\sum W(e) \leq \sum (W(e_1) + W(e_2)) \leq 2 \times W(M^*).$$

The below expression is valid,

$$\frac{1}{2} \times \text{bound} \leq W(M^*) = d_{M^*}^m(X, X') \leq \text{bound}.$$

### 5.8.3 Lower Bounding of Algorithm 5.4

Consider two point sets $X$ and $X'$ with $m \leq \min(|X|, |X'|)$, the function $d_{lw3}^m$ is to find the optimal selection of $m$ points from $X$ to minimize the approximate cost,

$$d_{lw3}^m(X, X') = \min_\tau \sum_{i=1}^{m_X} d_{MA}(X_{\tau,i}, X').$$

Suppose the subscripts of the selected $m$ vectors of the least cost of $d_{lw3}^m$ is $\tau^1$, and the selected points are denoted as $X_{\tau^1}$.

$$d_{lw3}^m(X, X') = \sum_{i=1}^{m_X} d_{MA}(X_{\tau^1,i}, X').$$
The points of $X_{\tau_1,i}$ are matched with those in $X'$ based on the bounding rectangle representations, therefore,

$$d_{MA}(X_{\tau_1,i}, X') \leq d_{M}^{X_{\tau_1,i}}(X_{\tau_1,i}, X').$$

The $\tau^1$ is a possible selection, the one to minimize $d_{iw3}^m$. It is true that,

$$d_{iw3}^m(X, X') = \sum_{i=1}^{m_X} d_{M}^{X_{\tau_1,i}}(X_{\tau_1,i}, X') \leq \min_{\tau} \sum_{i=1}^{m_X} d_{M}^{X_{\tau_1,i}}(X_{\tau_1,i}, X'). \quad (Eq. 5.3)$$

Suppose the selection of $m$ points to achieve the minimum of the distance $d_M^m$ is $\tau^2$, as the matching distance requires the matching to be feasible,

$$\sum_{i=1}^{m_X} d_{M}^{X_{\tau_2,i}}(X_{\tau_2,i}, X') \leq d_M^m(X, X'). \quad (Eq. 5.4)$$

The $\tau^2$ is just a possible selection and therefore, we can have the conclusion by combining Eq. 5.3 and Eq. 5.4,

$$d_{iw3}^m(X, X') \leq \min_{\tau} \sum_{i=1}^{m_X} d_{M}^{X_{\tau,i}}(X_{\tau,i}, X') \leq d_M^m(X, X').$$

As in Line 1 of Algorithm 5.4, we have the matrix $W_{mb_{br}}$, for $1 \leq i \leq m_X, 1 \leq j \leq m_{X'}$,

$$W_{mb_{br}}(i, j) = \left( \text{mindist}(MBR_i^X, MBR_j^{X'}) \right)^2.$$

The lower bound function $d_{iw3}^m$ is essentially a linear integer program with the object function:

$$\min_{x_{ij}} \sum_{i=1}^{m_X} \sum_{j=1}^{m_{X'}} W_{mb_{br}}(i, j)x_{ij}$$
subject to the constraints

\[ \sum_{i=1}^{m_X} \sum_{j=1}^{m_{X'}} x_{ij} = m, \]
\[ \sum_{j=1}^{m_{X'}} x_{ij} \leq |S_i^X|, \text{ for } 1 \leq i \leq m_X, \]
\[ x_{ij} \geq 0, \text{ for } 1 \leq i \leq m_X, 1 \leq j \leq m_{X'}. \]

The variable \( x_{ij} \) is the number of points selected from \( S_i^X \) to match with those in \( S_j^{X'} \).

Algorithm 5.4 is the greedy algorithm to solve the above problem directly. In each iteration, the current smallest entry is identified and as many points as possible are selected to match at this cost per selected point from \( S_i^X \). The value \( x_{ij} \) is computed as the value \( p \) in Line 5 of the algorithm for the selected cell, which is subject to the constraints of the linear integer formulation.
An object is denoted using a set of vectors. This new representation is powerful and useful in many real-world application domains. The similarity search problem over the point-set data has been studied in the previous chapter. The classification problem in this new data model is called *Multiple-Instance Learning* (MIL) [63] by the research community, in which an object is referred to as a bag and a feature vector corresponds to an instance. In this chapter, we study the classification problem and its application in the image database.

### 6.1 Introduction

In recent years, image repositories grow exponentially with the rapid advance in information technology. A wide range of content-based image applications have become very challenging because of the rapid expansion of representations and the volume of image data, and the complex nature of the high-level semantic meanings [124]. A significant amount of research efforts have been devoted to this interesting domain. In particular, image categorization is one of the major issues, that has attracted a great deal of attention. Image categorization is a systematic process that determines the class label for each image, from a set of pre-defined labels. This label thus reveals the semantic meaning of the image. Such labels are desirable in many applications.

In recent research, an image is usually characterized by a set of components. For example, an image can be segmented into a number of homogeneous regions [41], and one feature vector is extracted from each region to capture its visual properties, such as color, size, texture, shape, etc. Similarly, a set of salient regions or interest point descriptors [97] can be extracted for each image. Consequently, an image can be represented as a collection of feature vectors. Images with such representation can be categorized in two steps as follows. In phase 1, a
set of images with known labels are used in a learning process to design a classifier. This classifier is then used in phase 2 to label each image in the data set.

In the conventional multiple instance learning framework, the label of a bag is determined based on the labels of its instances [148]. Specifically, it is based on the following assumption: a bag is positive if some of its instances are positive; and a bag is negative if none of its instances are positive. There are generally two types of algorithms for classifying objects of multiple instances. One way is to learn from a training set a concept point [96] in the instance feature space, such that positive bags have some instances near this concept vector while all instances of the negative bags are far away from this concept point. With this concept point, its distance to the different bags in the data set can be used to classify the corresponding bags. Another classification approach is to adapt the supervised learning technique, support vector machine (SVM), to handle instance bags [148]. In this environment, the same assumption about positive bags is enforced in the constraints of the optimization problem in order to find the cutting hyperplane that separates positive and negative samples by the largest margin. A limitation of these existing approaches is due to aforementioned assumption about positive bags. Recent research has identified that this assumption may not suite for some application domains [41, 45], such as images and videos.

In this study, we propose a new learning technique whose design does not depend on the above assumption. In our method, each image (a bag of feature vectors) is converted into a single multi-dimensional vector; and therefore, the multiple-instance learning problem becomes a standard supervised learning problem. This can be achieved as follows:

(i) Instances (feature vectors) of the training images are clustered in the instance space to form instance clusters. Each cluster is given a label. Since each image has instances belonging to multiple instance clusters, it is associated with a number of cluster labels.

(ii) For each image category, we can treat each of the training images as an itemset (i.e., set of labels). Frequent itemset mining (counting co-occurring items) can then be applied
to discover frequent itemsets, each representing a set of labels that frequently co-occur in the images of this category.

(iii) Each frequent itemset is a set of cluster labels. If we replace each of these labels by a representative vector of this cluster, we can convert each itemset of labels into a set of feature vectors. These pattern vectors can be used to classify new images.

(iv) To facilitate learning, we consider a multidimensional space, of which the dimensionality is the number of distinct sets of pattern vectors derived in the last step. Each image, as a bag of feature vectors, is mapped to a point in this multidimensional space such that its coordinate in a given dimension is the degree of similarity between the feature vectors of this image and the pattern vectors corresponding to this dimension. With this mapping, the MIL problem is transformed into a standard supervised learning problem in this new space.

The contributions of the proposed method are as follows:

(i) We introduce a standard supervised learning technique for MIL problems.

(ii) We demonstrate the proposed technique through image categorization.

(iii) We propose to uncover common patterns of image bags in each category and design an effective technique to measure the similarity between two bags of multiple-instances.

(iv) Extensive experimental results are provided to illustrate the effectiveness of the new MIL technique.

The remainder of the chapter is organized as follows. Section 6.2 provides a survey of the related techniques. Our learning algorithm is discussed in detail in Section 6.3 and 6.4. The experimental results are reported in Section 6.5. Finally, we conclude the chapter in Section 6.6.
6.2 Related Works

In this section, we discuss related works in image categorization and Multiple-Instance Learning.

There have been many research efforts to develop low-level image descriptors, aiming to be able to effectively reflect the characteristics of images. The improved quality of image features can generally lead to a better categorization performance. Colors, textures, and shapes are considered representative components of images. Many advanced feature extraction algorithms have been proposed in the literature, for example, HSV color histogram, Gabor feature, Tamura feature and global texture feature \cite{59}. An image is characterized into a multi-dimensional feature vector and many categorization systems use different supervised learning algorithms in the core to perform the classification tasks, such as K-Nearest Neighbors Classifier, Neural Network \cite{29}, Support Vector Machine (SVM) \cite{34} and so on. The dimensionality of image features is usually very high. Different techniques have been proposed to obtain salient subspaces to achieve better classification accuracy and storage efficiency \cite{46}.

Recent proposals to denote an image as a bag of instances have attracted a significant amount of research interest \cite{40}. An image can be characterized using a set of segmented regions, salient regions or interest points \cite{97}. The bag representation can contain much more local information of an image compared to features generated by global extractors, however, this causes the associated learning problems much more challenging. Many previous MIL algorithms rely on the assumption that a positive bag has at least one instance labeled positive and none of the instances in a negative bag are positive. The Diversity Density (DD) algorithm \cite{96} aims to find a concept point in the instance feature space such that positive bags have some instances nearby while the negative instances are far away from the concept point. This algorithm is further integrated into the Expectation Maximization framework \cite{143} and extended to handle real-valued labels \cite{63}. The Citation-KNN \cite{131}
using the Hausdorff distance has been proposed to perform the classification task. The modified SVM, MI-SVM [20], can learn the label of a bag by treating instance labels as hidden variables. In [148], it has been proved that under the above mentioned assumption, the MIL problem is indeed related to the semi-supervised learning problem by enforcing the assumption as the constraints of the maximum margin problem. Accordingly, the Multi-Instance learning by Semi-Supervised Support Vector Machine (MissSVM) [148] is designed to solve the classification problem. There are also some other interesting works to deal with the MIL problem, like DD-SVM [42] and K-Means SVM [54].

The assumption that the above methods depend on has been shown ineffective in some application domains. In MILES [41], a coordinate in the feature space is defined for each bag. Each dimension of the coordinate of a bag corresponds to an instance from the training samples, whose value is related to the smallest distance between all instances of the bag and the instance in the training samples. The 1-norm SVM is utilized to solve the corresponding learning problem.

6.3 Overview of the Proposed Algorithm

Let \( \mathcal{X} \) be a set of images, which consists of \( \mathcal{K} \) categories and each class label is denoted as \( l_i \), for \( 1 \leq i \leq \mathcal{K} \). The \( i \)th image in \( \mathcal{X} \), \( X_i \) (\( 1 \leq i \leq |\mathcal{X}| \)), is represented by a set of feature vectors. \( X_i \) is called a bag and each of its vectors \( \vec{x}_{ij} \) is an instance:

\[
X_i = \{ \vec{x}_{i1}, \vec{x}_{i2}, \ldots, \vec{x}_{im_i} \},
\]

in which \( m_i \) is the number of instances contained in image \( i \) and the value \( m_i \) generally differs from image to image. Each \( \vec{x}_{ij} \) is a vector in the \( d \)-dimensional feature space, i.e., \( \vec{x}_{ij} \in \mathbb{R}^d \) and \( x_{ijk} \) is the \( k \)th attribute value in the \( j \)th instance of the \( i \)th image bag. There is a subset \( \mathcal{L} \) of images in \( \mathcal{X} \) and the rest are denoted as \( \mathcal{U} \). For each image \( X_i \) in \( \mathcal{L} \), its class label \( y_i \)
is already known, \( y_i \in \{l_1, l_2, \ldots, l_K\} \), while those in \( U \) are unlabeled. We would like to learn a function to properly predict the class labels for images in \( U \). The sets \( \mathcal{L} \) and \( U \) are called the training set and the test set respectively.

The major steps of our learning algorithm can be summarized as follows. For the given image set \( \mathcal{X} \), do the following:

(i) **Outlier Removal**: Outliers are detected and removed from all the instances of image bags because outliers do not carry much information on common patterns of each image category and may impede the subsequent data mining process.

(ii) **Instance-level Clustering**: We perform clustering over all the instances of the training samples; and each cluster is given a cluster label.

(iii) **Category Pattern Mining**: Each image is represented as a bag of cluster labels which correspond to the instance clusters its instances belong to. Data mining is performed to uncover patterns of cluster labels which co-occur frequently in the bags of each category.

(iv) **Pattern Feature Generation**: Each frequent label pattern of each category is converted into a bag of feature vectors as follows. For each label in the pattern, a representative vector is computed and consequently, the pattern is characterized by a set of feature vectors and the set is referred to as a pattern bag.

(v) **Bag-Pattern Distance Computation**: The distance metric between two bags of feature vectors can be defined as the sum of the pair-wise distances of their optimally matching vector pairs. The matching configuration is optimal in the sense that the sum is minimized. Using this distance function, each image bag is mapped into a point in a feature space, where each dimension corresponds to one of the distinct pattern bags. The coordinate of this point in a given dimension is computed as the distance between the image bag and the corresponding pattern bag.
Multi-Class Classification: Since each image is now represented as a feature vector, a standard supervised learning algorithm can be used for the image categorization task. In our study, we consider both the Multi-Layer Perceptron and Support Vector Machine for this learning step.

We discuss the above steps in more detail in the next section.

6.4 The Categorization Process in Details

6.4.1 Outlier Removal

There are totally $|\mathcal{X}|$ image bags in $\mathcal{X}$. All the instances $\vec{x}_{ij}$ in the whole dataset are placed bag-by-bag and instance-by-instance into one single set as follows:

$$\mathcal{X}_I = \{\vec{x}_{11}, \vec{x}_{12}, \ldots, \vec{x}_{1,m_1}, \vec{x}_{21}, \ldots, \vec{x}_{2,m_2}, \ldots, \vec{x}_{|\mathcal{X}|,1}, \ldots, \vec{x}_{|\mathcal{X}|,m_{|\mathcal{X}|}}\}.$$

In the first step, we would like to detect and eliminate outliers from the instance set. Outliers are data examples which appear differently from the rest of the dataset and can be regarded as noise. Outliers do not carry much valid information on common data patterns and may potentially impede the subsequent data analysis. The removal of outlier instances can properly generate a less disordered dataset that is beneficial to the classification task.

In the literature, there are a number of ways to define outliers [117]. The statistical approach assumes the given data follow some distribution model; and various statistical tests have been proposed to detect irregular examples which deviate significantly from the assumed model. As an example, Grubbs’ test can be used in conjunction with a normal distribution [98]. The distance-based approach includes effective non-parametric techniques, which examine the distances from a data point to its nearest neighbors. If a point is normal, its neighboring points should be close by; otherwise, the point is considered unusual because
its neighborhood is very sparse, i.e., nearest points are relatively far away. It has been shown that distance-based outliers are usually robust and practical [21] as this approach does not rely on any assumption about the data distribution. The algorithm in [24] uses simple pruning rules to detect distance-based outliers in near linear time with respect to the dimensionality and the size of the dataset. In our study, we choose this technique to extract abnormal examples out of the instance set $X_I$.

Given the set $X_I$, a score is defined for each instance vector and those having larger scores are identified as outliers. Formally, the score function of $\vec{x}_{ij} \in X_I$ is the average distance from $\vec{x}_{ij}$ to its $K_o$ nearest neighbors in $X_I$:

$$\text{score}(\vec{x}_{ij}) = \frac{\sum_{\vec{x} \in \text{NN}_{K_o}(\vec{x}_{ij})} d_2(\vec{x}_{ij}, \vec{x})}{K_o},$$

where $d_2$ is the Euclidean distance between the two $d$-dimensional vectors, i.e., for $\vec{x}_i, \vec{x}_j \in \mathbb{R}^d$,

$$d_2(\vec{x}_i, \vec{x}_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2},$$

and $\text{NN}_{K_o}(\vec{x}_{ij})$ is the set of $K_o$ nearest neighbors of $\vec{x}_{ij}$ in $X_I$ according to the Euclidean distance:

$$\text{NN}_{K_o}(\vec{x}_{ij}) = \{ \vec{x} \mid |\text{NN}_{K_o}(\vec{x}_{ij})| = K_o, \vec{x} \in (X_I - \{\vec{x}_{ij}\}),$$

$$\forall \vec{x}' \in (X_I - \text{NN}_{K_o}(\vec{x}_{ij}) - \{\vec{x}_{ij}\}), \mathcal{L}_2(\vec{x}_{ij}, \vec{x}) \leq \mathcal{L}_2(\vec{x}_{ij}, \vec{x}').$$

If a vector $\vec{x}_{ij}$ has a larger score, it is farther away from its nearest neighbors and therefore is more likely to be an outlier. This definition for mining the outliers is intuitive, fast, and robust [24]. The $N_o$ instance vectors in $X_I$ with the largest scores are selected as outliers. We denote this set as $O_{N_o}(X_I)$. 

131
With the set $O_{N_o}(X_I)$, we can examine each image bag $X_i$ in $X$ to identify and remove the outlier instances. The resulting bags $X'_i$, $1 \leq i \leq |X|$, are

$$X'_i = \{\vec{x}_{ij} \mid \vec{x}_{ij} \in X_i, \vec{x}_{ij} \notin O_{N_o}(X_I)\}.$$ 

It is possible that some $X_i$ has all its instances recognized as outliers; and therefore the corresponding $X'_i$ is empty. To handle these rare cases, we simply redefine $X'_i$ to be the original bag, $X_i$. Since we will use $X'_i$ in the rest of this chapter, $X_i$ is used to refer to $X'_i$ in the subsequent formulations for simplicity of notation.

### 6.4.2 Instance-level Clustering

In the second step of image categorization, we first cluster instance vectors of the training set $L$ into several groups. All feature vectors in $L$ are put into one single set as in the definition of $X_I$ in the last subsection:

$$L_I = \{\vec{x}_{ij} \mid X_i \in L\}.$$ 

In this step, the G-Means algorithm [74] is used to perform the clustering task. G-Means is a wrapper atop the conventional K-Means algorithm and does not require any tuning, as it can figure out the suitable number of clusters for the input data in a systematic way. The algorithm uses a statistical test with the hypothesis that data in a cluster follow a Gaussian distribution. Specifically, the data in an intermediate cluster are projected along the principal direction of this cluster and transformed into one-dimensional values. The Anderson-Darling statistical test can then be applied to the projection values to determine whether the hypothesis is accepted, i.e., whether the data in this cluster follow a Gaussian distribution. If an intermediate cluster appears non-Gaussian, it is split into two new smaller
clusters, and therefore, the number of clusters is increased by one. This procedure repeats until none of the clusters reject the Gaussian hypothesis.

Suppose for the instance set $L_I$, the number of clusters determined by the G-Means algorithm is $K_c$, and the corresponding cluster labels are $c_1, c_2, \ldots, c_{K_c}$. Let $f_c : L_I \rightarrow \{c_1, \ldots, c_{K_c}\}$ be the mapping from an instance vector $\vec{x}_{ij}$ to the label of the cluster it belongs to. For example, if $\vec{x}_{ij}$ is placed in the $k$th cluster by G-Means, then $f_c(\vec{x}_{ij}) = c_k$. The cluster label $c_k$ can be viewed as a compact description of the feature vector $\vec{x}_{ij}$.

Based on the clustering results, each image bag $X_i$ can be represented by the cluster labels of its instance vectors $\vec{x}_{ij}$. That is, each $\vec{x}_{ij}$ is replaced by its cluster label, the compact description of $\vec{x}_{ij}$. Formally, a cluster-label representation $S_i$ of $X_i = \{\vec{x}_{i1}, \vec{x}_{i2}, \ldots, \vec{x}_{im_i}\}$ is defined as:

$$S_i = \{f_c(\vec{x}_{i1}), f_c(\vec{x}_{i2}), \ldots, f_c(\vec{x}_{im_i})\}.$$

Based on the $S_i$ generated for all the images in $L$, we would like to find common label patterns for each category of images. This is discussed in details in the next subsection.

### 6.4.3 Category Pattern Mining

Each cluster label is an item in the set $\{c_1, c_2, \ldots, c_{K_c}\}$. The cluster-label representation $S_i$ of an image is a bag of items, in which an item can appear multiple times. We would like to uncover common label patterns for each image category. A pattern is a bag of items (cluster labels). It is supported by image $X_i$ if the pattern is a subset of the $S_i$. Consider an image category with three images, their cluster-label representations are as below:

$$S_1 = \{c_1, c_2, c_1, c_4, c_3\},$$

$$S_2 = \{c_1, c_1, c_2\},$$

$$S_3 = \{c_1, c_2, c_1, c_5\}.$$
We note that some cluster labels occur multiple times in the above cluster-label representations. This can happen because some instances of a given image bag may belong to the same instance cluster and therefore are assigned the same label. In the above example, it is interesting to observe that all three images have the pattern \( \{c_1, c_1, c_2\} \). This indicates that this pattern can potentially be used to characterize the image category. We are interested in applying data mining techniques to expose such label patterns. In particular, a standard frequent closed itemset mining algorithm \cite{33} can be employed for this purpose.

We need to pre-process the bags of labels before the standard frequent closed itemset mining algorithm can be applied. This is due to the fact that the standard technique is designed to deal with sets of items, not bags of items. Consequently, it is incapable of detecting multiple occurrences of an item in a pattern, and will miss out interesting patterns such as \( \{c_1, c_1, c_2\} \) for the above example. To address this problem, a superscript is added to each item of a cluster-label representation to index the occurrence of this item in the bag. Consider an instance vector \( \vec{x}_{ij} \) that belongs to the \( k \)th cluster, i.e., \( f_c(\vec{x}_{ij}) = c_k \). The corresponding item in \( S_i, c_k \) is replaced by \( c_k^m \) indicating that this item is the \( m \)th occurrence of the cluster label \( c_k \) in \( X_i \), i.e.,

\[
m = |\{\vec{x}_{il} \mid f_c(\vec{x}_{il}) = c_k, 1 \leq l \leq j - 1\}| + 1.
\]

For the above example, we have:

\[
S'_1 = \{c_1^1, c_2^1, c_1^2, c_4^1, c_3^1\},
\]
\[
S'_2 = \{c_1^1, c_2^2, c_1^2\},
\]
\[
S'_3 = \{c_1^1, c_2^1, c_1^2, c_5^1\}.
\]

After the pre-processing, the pattern \( \{c_1^1, c_2^1, c_2^1\} \) can be uncovered using the standard mining technique.
The cluster-label representation with superscripts $S'_i$ of each image is a set of items (called itemset) from the set $\{c^j_i\}$. For each image category $l_j$, frequent closed itemsets [33] can be mined from the corresponding representations, $\{S'_i \mid X_i \in \mathcal{L}, y_i = l_j\}$. Formally, a pattern $P^{\text{fc}}_i$ is denoted as $\{p^j_{i1}, p^j_{i2}, \ldots, p^j_{in_i}\}$ containing $n_i$ items, each of which is a cluster label with superscripts, i.e., $p^j_{ij} \in \{c^j_i\}$. The support of an itemset $P^{\text{fc}}_i$ in the $j$th image category is the percentage of the $S'_k$ in this category contains this itemset. That is,

$$support_j(P^{\text{fc}}_i) = \frac{|\{S'_k \mid X_k \in \mathcal{L}_j, S'_k \supseteq P^{\text{fc}}_i\}|}{|\mathcal{L}_j|},$$

where $\mathcal{L}_j$ contains all the bags in the $j$th category, i.e., $\mathcal{L}_j = \{X_i \mid X_i \in \mathcal{L}, y_i = l_j\}$.

In our example, the support of $\{c^1_1, c^1_2\}$ is 100%, that of $\{c^1_1, c^2_1, c^1_3\}$ is 100%, and that of $\{c^1_1, c^1_3\}$ is 33.3%. If an itemset has a support in the $j$th category no smaller than a threshold $T_s$, then this itemset is considered frequent in this category. An itemset is closed if all of its supersets have a smaller support. Hence, $\{c^1_1, c^2_1, c^1_2\}$ is a frequent closed itemset with $T_s = 50\%$ while $\{c^1_1, c^1_3\}$ is not. We apply a standard itemset mining package [31] to determine all the frequent closed itemsets for each image category, denoted as

$$\mathcal{P}^{\text{fc}} = \{(P^{\text{fc}}_i, b_i) \mid 1 \leq b_i \leq \mathcal{K}, \text{support}_{b_i}(P^{\text{fc}}_i) \geq T_s, P^{\text{fc}}_i \text{ is closed}\},$$

in which each entry indicates that $P^{\text{fc}}_i$ is frequent and closed in category $b_i$ ($1 \leq b_i \leq \mathcal{K}$).

The superscripts that index the multiple occurrences of cluster labels in a bag are only used in the itemset mining process discussed above. These superscripts are insignificant for the subsequent computations, and can be safely ignored in the following discussions for simplicity of notation. Thus, we have $P^{\text{fc}}_i = \{p^j_{i1}, \ldots, p^j_{in_i}\}$, where $p^j_{ij} \in \{c_i\}$. Accordingly, we can derive a set of patterns $\mathcal{P}^{\text{c}} = \{(P^{\text{c}}_i, b_i)\}$. 
6.4.4 Pattern Feature Generation

The preceding steps transform each image bag of feature vectors into a bag of cluster labels, and then perform data mining to uncover frequent patterns for each image category. Each pattern is a set of items from the set of cluster labels. In this subsection, we convert each pattern bag of each category into a bag of feature vectors.

For each image category, we first compute a representative $d$-dimensional vector for each relevant cluster label as follows. For each image bag $X_k$ in the $j$th category, we define a bag $Y_{j,c_i}^k$ which consists of only instance vectors belonging to cluster $c_i$, i.e.,

$$Y_{j,c_i}^k = \{ \vec{x}_{kl} \mid X_k \in \mathcal{L}_j, f_c(\vec{x}_{kl}) = c_i \}.$$

Since some of the images might not have any instance vector from cluster $c_i$, some of the sets as defined above might be empty. We define $\mathcal{Y}_{j,c_i}$ as the set of non-empty $Y_{j,c_i}^k$ sets as follows:

$$\mathcal{Y}_{j,c_i} = \{ Y_{j,c_i}^k \mid Y_{j,c_i}^k \text{ is not empty}\}.$$

Thus, $\mathcal{Y}_{j,c_i}$ is a set of bags. It contains all the instances that are grouped into cluster $c_i$ in $\mathcal{L}_j$, where $1 \leq i \leq K_c, 1 \leq j \leq K$. We can now compute a $d$-dimensional vector to represent a cluster label $c_i$ in category $j$, i.e., $\vec{v}_{j,c_i} \in \mathbb{R}^d$, as follows:

$$\vec{v}_{j,c_i} = \arg \min_{\vec{v} \in \mathbb{R}^d} \sum_{Y_{j,c_i}^k \in \mathcal{Y}_{j,c_i}} \min_{\vec{x}_{kl} \in Y_{j,c_i}^k} d_2(\vec{v}, \vec{x}_{kl}).$$

The optimal vector $\vec{v}_{j,c_i}$ has the smallest distance to some instance of each bag in $\mathcal{Y}_{j,c_i}$, which can be computed using Algorithm 6.1. This algorithm starts with an initial vector whose coordinates are set to 1, and iteratively refines this vector to eventually obtain the representative vector for the instance cluster in consideration. This algorithm guarantees to
find an optimal representative vector in the sense that a local minima of the above objective can always be achieved. A proof of this property is given in Section 6.7.

Algorithm 6.1 Compute optimal vector $v^{j,c_i}$

**Require:** the set of bags $Y^{j,c_i}$ ($1 \leq i \leq K_c, 1 \leq j \leq K$).

**Ensure:** optimal instance vector $v^{j,c_i}$.

1. Start with an initial $d$-dimensional instance vector $v = \langle 1, \ldots, 1 \rangle \in \mathbb{R}^d$.
2. For each $Y_k^{j,c_i} \in Y^{j,c_i}$, determine a vector $x^{j,c_i} \in Y_k^{j,c_i}$ which is closest to the vector $v$, and put them into one single set $S$,

$$S = \bigcup_{Y_k^{j,c_i} \in Y^{j,c_i}} \arg\min_{x^{j,c_i} \in Y_k^{j,c_i}} d_2(x^{j,c_i}, v).$$

3. Compute $v = \langle v_1, \ldots, v_d \rangle$ to be the mean vector of the set $S$,

$$v_i = \frac{1}{|S|} \sum_{x \in S} x_i,$$

in which $|S|$ is the cardinality of $S$, i.e., the number of vectors in the set.

4. Repeat steps 2 and 3 until convergence, and $v^{j,c_i}$ is the converged $v$.

After deriving all $v^{j,c_i}$ ($1 \leq i \leq K_c, 1 \leq j \leq K$), we compute a bag of vectors to represent each frequent closed itemset in the pattern set $P^c$. Each entry denotes a set of

$$P_i^c = \{ p_{i,1}^c, \ldots, p_{i,n_i}^c \}$$

supported by the $b_i$-th image category. Accordingly, a bag of $d$-dimensional vectors $P_i$ is defined for each $P_i^c$ by replacing $p_{i,j}^c$ with the corresponding vector in $\{ v^{j,c_i} \}$,

$$P_i = \{ v^{j_1,p_{i,1}^c}, \ldots, v^{j_{n_i},p_{i,n_i}^c} \}.$$  

In this way, each pattern $P_i^c$ of cluster labels is now instantiated by a vector bag $P_i$.

### 6.4.5 Bag-Pattern Distance Computation

An image bag is represented by a bag of $m_i$ instances $X_i = \{ \bar{x}_{i,1}, \ldots, \bar{x}_{i,m_i} \}$; and a pattern bag corresponds to a set of $n_j$ instances $P_j = \{ \bar{p}_{j,1}, \ldots, \bar{p}_{j,n_j} \}$. Generally, $X_i$ and $P_j$ may have
different numbers of vectors, that is \( m_i \neq n_j \). To facilitate similarity computation, we would like to pair the instances between these two bags. We can find \( M \) pairings such that each of the \( M \) instances in \( X_i \) is paired with a distinct instance in \( P_j \), in which \( M = \min(m_i, n_j) \).

The optimal bijective mapping leads to the minimal sum of the distances of the paired instances; and the similarity between the two bags can be defined accordingly based on the mapping. This optimal matching can be formulated as an assignment problem \([112, 45]\).

Mathematically speaking, let \( N = \max(m_i, n_j) \) and we pad \( X_i \) and \( P_j \) with ‘virtual instance vectors’ such that the two bags have the same cardinality, i.e., the same number \( N \) of instances. Each virtual instance vector is denoted as \( \vec{0}^v \in \mathbb{R}^d \), which is not a real vector, as it has a zero distances to any real vector in \( \mathbb{R}^d \), i.e., for any \( \vec{x} \in \mathbb{R}^d \), we have,

\[
d_2(\vec{0}^v, \vec{x}) = 0.
\]

The padded version of \( X_i \) is denoted as \( X^p_i \), in which \((N - m_i)\) virtual vectors are appended to \( X_i \). Similarly we can have the padded \( P^p_j \) for \( P_j \). That is,

\[
X^p_i = \{\vec{x}_i, \ldots, \vec{x}_{i,m_i}, \vec{0}^v, \ldots, \vec{0}^v\}_{N-m_i}, \text{ and } P^p_j = \{\vec{p}_j, \ldots, \vec{p}_{j,n_j}, \vec{0}^v, \ldots, \vec{0}^v\}_{N-n_j}.
\]

There exists a bijective mapping \( \tau : \vec{x} \in X^p_i \rightarrow \vec{p} \in P^p_j \), from instances of \( X^p_i \) to those of \( P^p_j \). For each \( \vec{x} \in X^p_i \), its correspondence in \( P^p_j \) is \( \tau(\vec{x}) \). The optimal mapping \( \tau \) minimizes the sum of paired distances,

\[
d_o(X^p_i, P^p_j) = \min_{\tau} \sum_{\vec{x} \in X^p_i} d_2(\vec{x}, \tau(\vec{x})).
\]

The above formulation is essentially an assignment problem, and can be solved efficiently using linear programming \([112]\). As the padded virtual vector \( \vec{0}^v \) has a zero distance to any vector, the distance \( d_o \) between \( X^p_i \) and \( P^p_j \) is the minimal sum of distances of the \( M \) optimally
paired instances between $X_i$ and $P_j$. Note that, if an image bag has fewer instances than
the pattern bag to be compared, this means that the image does not have correspondences
to some of the instances of the pattern and this may tend to underestimate the distance
between the two bags. To address this issue, we introduce an adjustment ratio $\frac{n_j}{m_i}$ in the
similarity computation. Accordingly, we define the (dis)similarity of $X_i$ with regarding to
the pattern $P_j$ as below,

$$
d_M(X_i \mid P_j) = \begin{cases} 
  d_o(X_i^p, P_j^p) & \text{if } m_i \geq n_j, \\
  \frac{n_j}{m_i} d_o(X_i^p, P_j^p) & \text{if } m_i < n_j.
\end{cases}
$$

To give an example of similarity computation, let us consider the following two bags:

$$
X_1 = \{\langle 1, 2 \rangle, \langle 2, 0 \rangle\} \text{ and } X_2 = \{\langle 3, 1 \rangle, \langle 1, 3 \rangle, \langle 2, 3 \rangle\}.
$$

We observe that $X_1$ has one fewer instance than $X_2$. In this case, $N = 3$, and we have
$X_i^p = \{\langle 1, 2 \rangle, \langle 2, 0 \rangle, \vec{0}^p\}$ and $X_2^p = X_2$. The optimal bijection $\tau$ between these two bags
maps $\vec{x}_{11}$ and $\vec{x}_{12}$ to $\vec{x}_{22}$ and $\vec{x}_{21}$, respectively; and $\vec{x}_{23}$ is mapped to $\vec{0}^p$. The $d_o$ distance is
computed as follows:

$$
d_o(X_1^p, X_2^p) = \sqrt{(1 - 1)^2 + (2 - 3)^2 + (0 - 1)^2} + 0 = 2.414.
$$

According to the definition, the $d_o$ distance is symmetric, while the $d_M$ distance is not due
to the adjustment ratio. If $X_1$ is a pattern, then $d_M(X_2 \mid X_1) = 2.414$. On the other hand, if $X_2$ is a pattern, then $d_M(X_1 \mid X_2) = \frac{3}{2} \times 2.414 = 3.621$.

As discussed before, a set of common patterns $P = \{P_i\}$ are derived based on the training
set $\mathcal{L}$. For each image bag $X_i$ in the dataset $\mathcal{X}$, we can define a feature vector, with $|P|$
values, each corresponding to the distance from $X_i$ to the pattern $P_j$,

$$\vec{f}_i = \langle f_{i1}, f_{i2}, \ldots, f_{i|P|} \rangle,$$

in which $f_{ij} = d_M(X_i \mid P_j)$. In this way, bags of different numbers of instances are transformed into feature vectors of the same dimensionality, and standard classification methods can be applied, as explained in the next subsection.

### 6.4.6 Multi-Class Classification

In our study, we have used two different standard algorithms to learn a model to classify different categories of images, namely Multi-Layer Perceptron (MLP) [29] and Support Vector Machine (SVM) [34]. We used the public available implementations, Weka [11] and LIBSVM [8, 39].

A multi-layer perceptron is a layered network of neurons, which defines a mapping of the input data onto desired outputs. The nodes of the consecutive layers are fully connected as a bipartite graph. Each link is associated with a tunable weight and each node corresponds to a sigmoid activation function. The gradient descent is used to train a network of weights in a backpropagation fashion for an optimal approximation to the desired outputs.

Support vector machine is a maximum margin classifier. In SVM, different kernel functions are applied to map the input data points into a very high dimensional space. After that, SVM is to find a cutting hyperplane to separate the relevant examples from those irrelevant with the largest margin. The instance vectors which are on or close to the separating boundary are called support vectors. SVM is inherently a binary classifier and can be extended for multi-class tasks in the one-versus-one fashion.
6.5 Categorization Performance

6.5.1 Experimental Settings

We evaluated the performance of our proposed categorization method against other state-of-the-art techniques on the public available dataset of 2,000 images [3] from the COREL repository. The image set contains 20 different categories of 100 images each. The category names are listed in Table 6.1. Each image is in JPEG format of size $384 \times 256$ or $256 \times 384$. An image is segmented into homogeneous regions using the modified K-Means algorithm, and each region is represented by a nine-dimensional feature vectors characterizing color, texture and shape properties of the region. Consequently, each image is represented as a bag of instance vectors in $\mathbb{R}^9$. Table 6.1 shows the average number of instances per bag for each category. We used the processed data [3] in the subsequent processing. The original data were used as two datasets. The first one (1000-Image) contains the first ten categories (from ID 0 to 9), and the second one (2000-Image) uses all the classes.

In our proposal, we used the Orca package [13] for outlier detection, and the number of nearest neighbors $K_o$ used to detect outliers was set to be 5, and the top $N_o = 100$ instances were identified as outliers and eliminated. For the third step, the Apriori package [6] was used to mine the frequent closed patterns with the support threshold $T_s = 20\%$. All the other processing codes were written in MATLAB, to transform an image bag to a feature vector. In the last step, we used either MLP or SVM to learn the classification model, and correspondingly, our methods are respectively denoted as BP-MLP and BP-SVM. In BP-MLP, the neural network in the Weka software [11] was used with the default parameters. In BP-SVM, the LIBSVM library [8] was utilized. We chose the C-SVC with the Radial basis kernel function. The parameters, cost $C$ and gamma $\gamma$, were selected using grid search ($C = 2^{-5}, 2^{-3}, \ldots, 2^{15}$, $\gamma = 2^{-15}, 2^{-13}, \ldots, 2^{3}$) in a twofold cross-validation on the training set. The values which gave the minimum twofold cross-validation error were chosen to carry...
<table>
<thead>
<tr>
<th>ID</th>
<th>Category Name</th>
<th>Instances/bag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>African people and villages</td>
<td>4.84</td>
</tr>
<tr>
<td>1</td>
<td>Beach</td>
<td>3.54</td>
</tr>
<tr>
<td>2</td>
<td>Historical building</td>
<td>3.10</td>
</tr>
<tr>
<td>3</td>
<td>Buses</td>
<td>7.59</td>
</tr>
<tr>
<td>4</td>
<td>Dinosaurs</td>
<td>2.00</td>
</tr>
<tr>
<td>5</td>
<td>Elephants</td>
<td>3.02</td>
</tr>
<tr>
<td>6</td>
<td>Flowers</td>
<td>4.46</td>
</tr>
<tr>
<td>7</td>
<td>Horses</td>
<td>3.89</td>
</tr>
<tr>
<td>8</td>
<td>Mountains and glaciers</td>
<td>3.38</td>
</tr>
<tr>
<td>9</td>
<td>Food</td>
<td>7.24</td>
</tr>
<tr>
<td>10</td>
<td>Dogs</td>
<td>3.80</td>
</tr>
<tr>
<td>11</td>
<td>Lizards</td>
<td>2.80</td>
</tr>
<tr>
<td>12</td>
<td>Fashion models</td>
<td>5.19</td>
</tr>
<tr>
<td>13</td>
<td>Sunset scenes</td>
<td>3.52</td>
</tr>
<tr>
<td>14</td>
<td>Cars</td>
<td>4.93</td>
</tr>
<tr>
<td>15</td>
<td>Waterfalls</td>
<td>2.56</td>
</tr>
<tr>
<td>16</td>
<td>Antique furniture</td>
<td>2.30</td>
</tr>
<tr>
<td>17</td>
<td>Battle ships</td>
<td>4.32</td>
</tr>
<tr>
<td>18</td>
<td>Skiing</td>
<td>3.34</td>
</tr>
<tr>
<td>19</td>
<td>Desserts</td>
<td>3.65</td>
</tr>
</tbody>
</table>

Table 6.1: Twenty image categories and the average number of instances per bag for each category

out the classification task on the test set. All the experiments were performed on a computer with Core 2 Duo T7200 2GHz and 1 GB RAM running Microsoft Windows XP Professional SP2.

### 6.5.2 Categorization Performance

For each dataset, the whole set was divided into 10 folds. In each run, one fold was used as the test samples and the others were for training. Below we discussed the experiments on the 1000-Image set in detail. The instance level clustering was executed on all the instances in the training samples. On average, the G-Means divided the instances into 68 groups. The common patterns were further mined out for each category in the training set. Table 6.2 reports the average number of patterns and also their average cardinality (the number
of instances per pattern) generated for each category of each run. It can be seen that the number of frequent patterns greatly differ from category to category. Generally, the number of patterns of an image category corresponds to the complexity of the visual appearing of this semantic class. It is interesting that the 5th concept (images of dinosaurs) is the simplest one, only having 3 patterns of 1.67 instances each. The 4th category (buses) appears the most complex, which has 62.2 common patterns. With regard to the derived patterns, a feature vector was computed for each image, which were the distances of the bag to each pattern. The dimensionality of transformed feature vectors was equal to the number of patterns, which on average was 231.2.

The confusion matrix of BP-MLP on the 1000-Image set is shown in Table 6.4. Each row presents the average percentage of images of the category (in this row) classified into each of the 10 classes (corresponding to different columns). The values along the diagonal (in bold) correspond to the classification accuracy of each concept and off-diagonal cells are related to classification errors. Generally, our BP-MLP performed well in this multi-class task, especially for Category 3, 4, 6 and 7, with less than 10% classification error. Note that the two largest classification mistakes occurred between Category 1 (beach) and 8 (mountains and glaciers), and the corresponding entries are highlighted with underlined numbers. 18.9% of images in Category 1 were recognized as Category 8 by mistake and 12.3% of Category 8 were mis-classified into Category 1. This observation is consistent with others’ work [41,148], as these two class contain visually similar regions, corresponding to mountain, river, lake and ocean.

The average categorization accuracy of our proposals BP-MLP and BP-SVM are reported in Table 6.3. The table also includes the results of some algorithms reported in the literature, such as MissSVM [148], MILES [41], DD-SVM [42], MI-SVM [20] and K-Means SVM [54]. This table clearly indicates that our algorithm to transform a bag into a feature vector is effective, and the bag-pattern distances are meaningful which can be used to differentiate the image categories. Both of our methods outperform the compared ones. The BP-MLP
Table 6.2: The number of patterns and the average number of instances per pattern for each category on 1000-Image

<table>
<thead>
<tr>
<th>Category (ID: 0 - 9)</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num of Patterns</td>
<td>23.0</td>
<td>10.2</td>
<td>12.1</td>
<td>62.2</td>
<td>3.0</td>
<td>14.1</td>
<td>30.2</td>
<td>24.0</td>
<td>11.5</td>
<td>40.9</td>
</tr>
<tr>
<td>Instances/Pattern</td>
<td>1.20</td>
<td>1.00</td>
<td>1.22</td>
<td>1.83</td>
<td>1.67</td>
<td>1.01</td>
<td>1.60</td>
<td>1.60</td>
<td>1.00</td>
<td>1.35</td>
</tr>
</tbody>
</table>

Table 6.3: Average categorization accuracy of different techniques

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>1000-Image</th>
<th>2000-Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP-MLP</td>
<td>83.4</td>
<td>69.0</td>
</tr>
<tr>
<td>BP-SVM</td>
<td>82.9</td>
<td>68.8</td>
</tr>
<tr>
<td>MissSVM</td>
<td>78.0</td>
<td>65.2</td>
</tr>
<tr>
<td>MILES</td>
<td>82.6</td>
<td>68.7</td>
</tr>
<tr>
<td>DD-SVM</td>
<td>81.5</td>
<td>67.5</td>
</tr>
<tr>
<td>MI-SVM</td>
<td>74.7</td>
<td>54.6</td>
</tr>
<tr>
<td>k-means SVM</td>
<td>69.8</td>
<td>52.3</td>
</tr>
</tbody>
</table>

has better classification accuracy than the BP-SVM, though the MLP took much more time to learn the model than the SVM, as shown in Table 6.5. The BP-MLP took 1497.7 seconds to learn the neural network model to classify the COREL 1000 images dataset on average while the BP-SVM only took around 848.1 seconds. It is clear that the BP-SVM is superior to the BP-MLP in term of the computational efficiency. As shown in [41], the MILES technique is much faster than the DD-SVM. Our methods and MILES are similar as they all map the bags into a feature space. The dimensionality of the embedding of MILES is the number of instances of all the bags in the training set. This number is generally much larger than the number of patterns mined in our methods, which is the dimensionality of the feature space generated by our methods. Therefore, the embedding of MILES is of a much higher dimensionality and it generally takes much more time in training. Our proposals are computationally less expensive than MILES and DD-SVM in the training process.
Table 6.4: The confusion matrix of BP-MLP on 1000-Image

<table>
<thead>
<tr>
<th></th>
<th>Cat. 0</th>
<th>Cat. 1</th>
<th>Cat. 2</th>
<th>Cat. 3</th>
<th>Cat. 4</th>
<th>Cat. 5</th>
<th>Cat. 6</th>
<th>Cat. 7</th>
<th>Cat. 8</th>
<th>Cat. 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat. 0</td>
<td>79.2</td>
<td>2.0</td>
<td>6.1</td>
<td>0.0</td>
<td>0.8</td>
<td>7.2</td>
<td>0.2</td>
<td>3.2</td>
<td>1.1</td>
<td>0.2</td>
</tr>
<tr>
<td>Cat. 1</td>
<td>1.9</td>
<td>63.4</td>
<td>3.6</td>
<td>2.8</td>
<td>0.6</td>
<td>5.9</td>
<td>1.0</td>
<td>0.0</td>
<td>18.9</td>
<td>1.9</td>
</tr>
<tr>
<td>Cat. 2</td>
<td>6.4</td>
<td>3.5</td>
<td>70.5</td>
<td>3.4</td>
<td>1.0</td>
<td>6.3</td>
<td>1.3</td>
<td>0.0</td>
<td>6.1</td>
<td>1.5</td>
</tr>
<tr>
<td>Cat. 3</td>
<td>0.0</td>
<td>0.7</td>
<td>2.7</td>
<td>94.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.3</td>
<td>1.3</td>
</tr>
<tr>
<td>Cat. 4</td>
<td>0.0</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>99.4</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Cat. 5</td>
<td>5.4</td>
<td>1.8</td>
<td>5.4</td>
<td>0.0</td>
<td>2.7</td>
<td>74.6</td>
<td>0.0</td>
<td>1.9</td>
<td>7.3</td>
<td>0.9</td>
</tr>
<tr>
<td>Cat. 6</td>
<td>1.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>94.5</td>
<td>0.0</td>
<td>1.5</td>
<td>2.7</td>
</tr>
<tr>
<td>Cat. 7</td>
<td>1.9</td>
<td>0.0</td>
<td>1.1</td>
<td>0.0</td>
<td>0.0</td>
<td>1.1</td>
<td>0.9</td>
<td>95.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Cat. 8</td>
<td>0.0</td>
<td>12.3</td>
<td>7.3</td>
<td>1.4</td>
<td>0.0</td>
<td>3.7</td>
<td>0.0</td>
<td>0.0</td>
<td>75.3</td>
<td>0.0</td>
</tr>
<tr>
<td>Cat. 9</td>
<td>3.7</td>
<td>1.8</td>
<td>0.0</td>
<td>4.6</td>
<td>0.0</td>
<td>0.9</td>
<td>0.0</td>
<td>0.9</td>
<td>0.0</td>
<td>88.1</td>
</tr>
</tbody>
</table>

Table 6.5: Average elapsed time (in seconds) to train the classification models by the proposed techniques

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>1000-Image Dataset</th>
<th>2000-Image Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP-MLP</td>
<td>1497.7</td>
<td>5424.9</td>
</tr>
<tr>
<td>BP-SVM</td>
<td>848.1</td>
<td>3182.5</td>
</tr>
</tbody>
</table>

6.6 Summary

We have presented a novel learning algorithm for image categorization in which images are represented in bags of instances. Our approach does not try to learn the labels of instances, which are typically used to decide the label of a bag directly. Instead, we extract patterns which appear frequently in each image category. An image bag is embedded into a data point in the multi-dimensional space, which well characterizes the similarity between the image and every common pattern of an image category. Therefore, the MIL problem is formulated into a standard supervised learning problem. The experiments demonstrate that our embedding for bags of instances is effective, and the categorization accuracy is competitive to other state-of-the-art algorithms.
In this section, we would like to prove the correctness of Algorithm 6.1. The problem to be solved is to find an optimal vector $\vec{v}_{j,c_i}$ satisfying,

$$\vec{v}_{j,c_i} = \arg\min_{\vec{v} \in \mathbb{R}^d} \sum_{Y_{j,c_i} \in \mathcal{Y}_{j,c_i}} \min_{x_{kl} \in Y_{k,c_i}} d_2(\vec{v}, \vec{x}_{kl}),$$

in which, $\mathcal{Y}_{j,c_i}$ is a set of bags with regard to cluster $c_i$ and category $j$.

For a vector $\vec{v}$, and a mapping function $\Upsilon$, denote $\Upsilon(Y_{j,c_i}^k)$ to be the instance from the bag $Y_{j,c_i}^k$ which is mapped to vector $\vec{v}$. Therefore the optimal $\vec{v}_{j,c_i}$ also achieves the below optimization problem,

$$\vec{v}_{j,c_i} = \arg\min_{\vec{v}, \Upsilon} \sum_{Y_{j,c_i} \in \mathcal{Y}_{j,c_i}} d_2(\vec{v}, \Upsilon(Y_{j,c_i}^k)).$$

Therefore, we would like to find a vector to minimize

$$f(\vec{v}, \Upsilon) = \sum_{Y_{j,c_i}^k \in \mathcal{Y}_{j,c_i}} d_2(\vec{v}, \Upsilon(Y_{j,c_i}^k)). \quad \text{(Eq. 6.1)}$$

For a fixed vector $\vec{v}$, the optimal mapping of each bag $Y_{k,c_i}^j$ is to map $\vec{v}$ to the closest one among all the instances in $Y_{k,c_i}^j$, i.e.,

$$\Upsilon(Y_{k,c_i}^j) = \arg\min_{\vec{x} \in Y_{k,c_i}^j} d_2(\vec{x}, \vec{v}).$$

For a fixed mapping $\Upsilon$, the optimal vector is the centers of all the matched instances, i.e.,

$$\vec{v} = \frac{1}{|\mathcal{Y}_{j,c_i}|} \sum_{Y_{j,c_i}^k \in \mathcal{Y}_{j,c_i}} \Upsilon(Y_{j,c_i}^k).$$
Algorithm 6.1 takes the iterative approach to reach a local minimum of the objective function defined in Eq. 6.1. It starts with an initial guess of the vector. Each run of Step 2 of the algorithm is to find the matched instances with regard to the current vector. This guarantees to reduce the objective. In Step 3, the vector to be computed is updated as the centroid of the matched instances, which certainly decreases the objective value. Therefore Algorithm 6.1 is sure to have the objective value smaller and smaller. Because there are only a finite number of instances in $Y_{j,i}^{j,e}$, there only exists a finite number of mapping, and the objective function defined in Eq. 6.1 is lower-bounded. Overall, Algorithm 6.1 guarantees to converge and the derived optimal vector gives a local minimum of the objective function.
CHAPTER 7: DISCUSSIONS

The dissertation has covered a number of different research topics. This chapter summarizes the understandings and observations which have been reached in this research study.

7.1 Dimension Reduction and Manifold Learning

For high-dimensional data, its intrinsic dimensionality is much smaller. It is generally believed that the inherent structures of data (the manifold structures) actually reside in the low-dimensional space. Consequently, the original data can be projected into a lower-dimensional space. The embedding aggressively reduces the storage space and computational processing cost, while it may distort the relationships between data points (whether the points are near or far away in the original space), which are referred to as local and global structures of the data. Dimension reduction is the classic and broad topic. Manifold learning focus more on automatically discovering and preserving the structures of the data, which is of recent research interest. There have been a large number of proposals in the literature to reduce the dimensionality of the data.

One approach is to convert the reduction problem into the eigenvalue and eigenvector problem. Generally the technique is linear, robust and defined everywhere in the space, in which the original data are projected along a small number of salient directions determined by different criteria. Generally, these projection directions are orthogonal to each other, in order to guarantee no false dismissal in the query evaluation. The Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Locality Preserving Projection (LPP) and our proposal in Chapter 2, LGSPP belong to this category. The PCA and SVD tries to push the data points as far away from each other in the reduced space as possible. In other words, the total sum of pairwise distances of the points in the generated space is maximized in the PCA and SVD. In transformed space determined by the LPP, points are kept close to
their respective nearest neighbors. Technically speaking, it minimizes the sum of distances between each point and its neighbors while avoiding the collapse of the space. The proposed LGSPP can be considered as the marriage of PCA and LPP, which preserves the local and global structures.

The SubSpace Projection (SSP) in Chapter 4 is the generic framework for partition-based dimension reduction techniques. The dimensions are divided into subgroups, each of which form a subspace. The reduction is carried out in each individual subspace and a vector is decomposed along the diagonal line in the subspace. To determine an optimal dimension partition is important to achieve robust retrieval performances and this problem has been solved in the study by a greedy algorithm.

Many recent proposals in manifold learning are non-linear. Those techniques are derived based on different assumptions and data models. They work well in the synthetic datasets. However, generally there are still some discrepancies between the assumptions, models and the real-life data. Hence, the performances of non-linear techniques are still not satisfactory in practice.

The above line of reduction schemes are data dependent. It need to have some pre-processing over the data before computing the embedding. Basically the pre-processing analyzes the structures of the entire dataset, which can help determine the optimal transform adaptive to the given data. Instead, many data-independent techniques have their root in information theory. Significant components of a data point are extracted independently. A vector can be consider as a sequence of values, essentially signals. The signal is decomposed as a combination of fourier coefficients, wavelet coefficients or cosine coefficients of different frequencies. The techniques are respectively called Discrete Fourier Transform (DFT), Discrete Wavelet Transform (DWT) and Discrete Cosine Transform (DCT). Besides the signal processing approach, the Vector Approximation file (VA file) technique executes the compression on the original data into a much smaller file and linear scan on the compressed data.
7.2 Similarity Search in Vector Space

In the conventional model, objects are represented as points in the vector space. The similarity search problem is to identify objects which are near the query object in the vector space, determined by some distance function. In vector space, the Euclidean distance is the popular metric. Feature selection and weighting and normalization schemes have been developed to improve the retrieval accuracy. Data normalization can remove the effects of translation, scaling and inverse scaling. Mahalanobis distance leverages a matrix to normalize the data so that the transformed data have a regular sphere shape and usually the normalization matrix is the covariance structure of the data. Some other metrics are developed in the past, like Kullback-Leibler divergence. The naive approach to evaluate a similarity query is to perform a linear scan over the dataset and compute distances between the query object and all the data objects. There are different ways to accelerate the query evaluation.

The classic approach is to utilize the spatial indexing techniques. These techniques essentially are space partitioning, either partition the data space into (overlapped or non-overlapped) bounding rectangles or sphere rings. Then for a given query point, consider a rectangle (ring), two important values are defined, the minimum possible distance and the maximum possible distance. The minimum possible distance is the smallest distance between all possible points contained in the rectangle (ring) and the query point. The maximum possible distance can be defined similarly. These two values can be used to quickly prune the irrelevant rectangles (rings). In general, a query only intersects with a small number of rectangles (rings). In the \( k \)-nearest-neighbor search, the algorithm can start with the rectangle (ring) which has the smallest minimum possible distance and gradually explore the space. However, the spatial access method usually works with data of smaller than 20 or 30 dimensions.

The minimum possible distance can be considered as lower-bounding approximation. The minimum possible distance is the lower bound of all distances between any points in the
rectangle (ring) and a point or another rectangle (ring). This lower-bound distance can help determine whether it is necessary to examine the objects contained in the corresponding rectangle (ring) and compute the exact distances. However, lower bounding is a board concept, which is much more than the minimum possible distance in the spatial indexing. Lower-bounding guarantees no-false-dismissal in the query results. It has been widely used in similarity search in complex data models and sophisticated distance functions.

Many salient distance functions have been developed to measure the similarity degree between sequences, for example the edit distance, dynamic time warping, longest common subsequence and so on. Generally, it is infeasible to partition the data space of sequences and so far there does not exist general-purpose indexing schemes for sequences. The lower-bounding approximation distance between two sequences is designed to be always no larger than the exact distance and this can be used to avoid the unnecessary evaluations of the exact distance function.

The early-stop mechanism is another way to accelerate the distance computation, but it can only be leveraged if the distance function satisfies the property: in the process to compute the distance, the intermediate value increases gradually step by step. Many distance functions do not hold this property. For those which have this trait, the distance evaluation can be terminated earlier before the exact distance value is derived. As soon as the intermediate distance value exceeds the current nearest neighbor candidates’ distance, then the computation can stop. This property has been exploited in the matching distance as discussed in Chapter 5.

Approximate similarity search attracts recent interest. It executes the queries and identifies the nearest neighbors with a high probability. This line of techniques usually utilize random projection and hashing. The hashing can aggressively map a high-dimensional vector into a single value.
7.3 Data Clustering

Data clustering is to automatically partition data objects into meaningful groups. Similar objects are put into the same cluster. The input of the clustering problem can take two forms. In the first form, objects are points and there is a distance function between the points, which is common in many applications. In the other form, it may not have access to the actual content of the objects but it has the pairwise matrix recording the distances between every two objects.

There are a large number of clustering algorithms in the vector space. The partition-based methods include K-Means, K-Medoids and PAM. The hierarchical/agglomerative algorithm can split (merge) the data in the top-down (bottom-up) fashion. Most of the density-based clustering algorithms are developed to identify dense regions in the database approach, for example DBSCAN and DENCLUE, which put more focus on the efficiency of the algorithm design and try to reduce the number of database pass. The Expected-Maximization (EM) algorithm maximizes the likelihood probability to generate the data based on the model of the clustering result.

Spectral clustering and graph-theoretic clustering algorithms take the pairwise similarity matrix as the input. They either maximize the overall similarity between points in each cluster or maximizes the dissimilarity of points between different clusters. Graph clustering has a strong relationship with the Kernel K-Means, which can be considered to be the K-Means algorithm in the kernel space. The belief propagation (via message passing), non-negative matrix factorization and correlation clustering are some recent developments in the clustering algorithm.

As discussed in the previous sections, data generally exhibit different correlation structures in different regions in the space. Many clustering proposals are coupled with the dimension reduction, feature selection and feature weighting schemes. These techniques are
called subspace clustering, which aim to discover the meaningful clusters residing in the subspaces. Our locally weighted clustering algorithm in Chapter 3 belongs to this category.

The above discussed clustering techniques are unsupervised algorithms. The recent trend is to leverage partial user knowledge to further improve the clustering accuracy. Instance-level constraints specify the semantic relations of a pair of points, whether the two points belong to the same cluster. These constraints can help to find better subspaces and also enforce the cluster results to better satisfy the constraints. Basically, the semi-supervised algorithms either penalize the constraint violations or guide the clustering assignment stage, as our proposal in Chapter 3.

In the clustering research, the problem to determine the number of clusters in a dataset is also important. The conventional information-theoretic techniques include Bayesian information criterion, Akaike information criterion, minimum description length, minimum message length and normalized entropy criterion. The statistical hypothesis is used to test whether a cluster follow the assumed data distribution and split the cluster into two accordingly. There are also some research works in self-tune spectral clustering. Some additional studies have been done in the consensus clustering to produce robust results and hypergraph clustering which partitions the edge graphs.

### 7.4 Design of Distance Metrics

The Euclidean distance is the robust distance metric in the data model which assumes a class of data are generated by the Gaussian distribution. The edit distance is the minimum number of operations (addition, deletion or modification) which change one string of characters into another. The dynamic time warping has the similar favor to the edit distance, which aligns two data sequences. In the computation of the dynamic time warping, sequences can be stretched locally to minimize the sum of differences after the transformed sequences. The Earth Mover Distance (EMD) measures the similarity of data histograms, which are
bins of counters. The EMD distance uses additional weights to adjust the importance of
the matchings between two bins. The EMD distance is formulated into the transportation
problem and optimizes the sum of weighted differences between the bins.

The string of characters and the sequence of values have order, i.e., characters and values
are ordered in the underlying axis. In the point-set data model, a set of points have no
defined order. The matching distance studied in Chapter 5 minimizes the sum of distances
of two point sets allowing flexible one-to-one matchings. The matching distance is formulated
into the assignment problem.

Matching is the key ingredient to design distance functions for data models beyond the
single point model. The robust distance is the minimum sum of differences between the
matched elements under all possible feasible matchings. Each different model defines its own
set of allowed matchings, which best suite with the application domain.

To design a distance function, it is also good to guarantee it is indeed a metric, which is
non-negative, identity of indiscernibles, symmetric and holds the triangle inequality rule. If
the function is a metric, then many well-established methods can be utilized to accelerate
the query processing and many formal tools can be used to study this distance metric.

7.5 Multiple Instance Learning

Many multiple instance learning algorithms are built based on the assumption that there
exists an instance vector representing a concept. A positive set has some points close to the
concept vector and all points of a negative set are far away from the concept instance. The
real-world datasets may not well fit this assumption. Our proposal in Chapter 6 does not
rely on this assumption. Instead, it develops the automatic way to extract the patterns of
each category and transform an object (a point set) into a vector of values indicating the
degree of similarity between the object and each individual pattern. Hence, a point set is
transformed into a vector and the multiple instance learning problem is converted into the
standard supervised learning problem. This study also well matches the observation that an image category exhibits heterogenous patterns, which can be mined systematically.
REFERENCES


the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 29–38, New York, NY, USA, August 2003. ACM.


160


