K-means-based Consensus Clustering: Algorithms, Theory and Applications

A Dissertation Presented

by

Hongfu Liu

to

The Department of Electrical and Computer Engineering

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Northeastern University
Boston, Massachusetts

2018
Dissertation Title: K-means-based Consensus Clustering: Algorithms, Theory and Applications

Author: Hongfu Liu

Department: Electrical and Computer Engineering

Approved for Dissertation Requirements of the Doctor of Philosophy Degree

Dissertation Advisor
Dr. Yun Fu
Signature Date

Dissertation Committee Member
Dr. Jennifer G. Dy
Signature Date

Dissertation Committee Member
Dr. Lu Wang
Signature Date

Department Chair
Dr. Srinivas Tadigadapa
Signature Date

Associate Dean of Graduate School:
Dr. Thomas C. Sheahan
Signature Date
To my family.
# Contents

<table>
<thead>
<tr>
<th>List of Figures</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>xi</td>
</tr>
<tr>
<td>Abstract of the Dissertation</td>
<td>xii</td>
</tr>
</tbody>
</table>

## 1 Introduction

1.1 Background .................................................. 1
1.2 Related Work ............................................... 2
1.3 Dissertation Organization ................................. 3

## 2 K-means-based Consensus Clustering

2.1 Preliminaries and Problem Definition .................. 4
2.1.1 Consensus Clustering .................................. 5
2.1.2 K-means Clustering .................................. 6
2.1.3 Problem Definition ................................... 8
2.2 Utility Functions for K-means-based Consensus Clustering .......................... 9
2.2.1 From Consensus Clustering to K-means Clustering .. 9
2.2.2 The Derivation of KCC Utility Functions ............. 11
2.2.3 Two Forms of KCC Utility Functions ................. 13
2.3 Handling Incomplete Basic partitions .................. 15
2.3.1 Problem Description ................................ 15
2.3.2 Solution ............................................. 16
2.4 Experimental Results ..................................... 17
2.4.1 Experimental Setup ................................ 18
2.4.2 Clustering Efficiency of KCC ......................... 19
2.4.3 Clustering Quality of KCC .......................... 21
2.4.4 Exploration of Impact Factors ....................... 22
2.4.5 Performances on Incomplete Basic partitions ........ 29
2.5 Concluding Remarks ...................................... 30
3 Spectral Ensemble Clustering

3.1 Spectral Ensemble Clustering

3.1.1 From SEC to Weighted K-means

3.1.2 Intrinsic Consensus Objective Function

3.2 Theoretical Properties

3.2.1 Robustness

3.2.2 Generalizability

3.2.3 Convergence

3.3 Incomplete Evidence

3.4 Towards Big Data Clustering

3.5 Experimental Results

3.5.1 Scenario I: Ensemble Clustering

3.5.2 Scenario II: Multi-view Clustering

3.5.3 SEC for Weibo Data Clustering

3.6 Summary

4 Infinite Ensemble Clustering

4.1 Problem Definition

4.2 Infinite Ensemble Clustering

4.2.1 From Ensemble Clustering to Auto-encoder

4.2.2 The Expectation of Co-Association Matrix

4.2.3 Linear version of IEC

4.2.4 Non-Linear version of IEC

4.3 Experimental Results

4.3.1 Experimental Settings

4.3.2 Clustering Performance

4.3.3 Inside IEC: Factor Exploration

4.4 Application on Pan-omics Gene Expression Analysis

4.4.1 Experimental Setting

4.4.2 One-omics Gene Expression Evaluation

4.4.3 Pan-omics Gene Expression Evaluation

4.5 Summary

5 Partition-Level Constraint Clustering

5.1 Constrained Clustering

5.2 Problem Formulation

5.2.1 Partition Level Side Information

5.2.2 Problem Definition

5.2.3 Objective Function

5.3 Solutions

5.3.1 Algorithm Derivation

5.3.2 K-means-like optimization

5.4 Discussion

5.4.1 Handling Multiple Side Information

5.4.2 PLCC with Spectral Clustering
5.5 Experimental results

5.5.1 Experimental Setup

5.5.2 Effectiveness and Efficiency

5.5.3 Handling Side Information with Noises

5.5.4 Handling Multiple Side Information

5.5.5 Inconsistent Cluster Number

5.6 Application to Image Cosegmentation

5.6.1 Cosegmentation

5.6.2 Salincy-Guided Model

5.6.3 Experimental Result

5.7 Summary

6 Structure-Preserved Domain Adaptation

6.1 Unsupervised Domain Adaptation

6.2 SP-UDA Framework

6.2.1 Problem Definition

6.2.2 Framework

6.2.3 SP-UDA for Single Source Domain

6.2.4 SP-UDA for Multiple Source Domains

6.2.5 K-means-like Optimization

6.3 Experimental Results

6.3.1 Experimental Settings

6.3.2 Object Recognition with SURF Features

6.3.3 Object Recognition with Deep Features

6.3.4 Face Identification

6.4 Summary

7 Conclusion

Bibliography

A Appendix

A.1 Proof of Lemma 2.2.1

A.2 Proof of Lemma 2.2.2

A.3 Proof of Theorem 2.2.1

A.4 Proof of Proposition 2.2.1

A.5 Proof of Theorem 2.3.1

A.6 Proof of Theorem 3.1.1

A.7 Proof of Theorem 3.1.2

A.8 Proof of Theorem 3.2.1

A.9 Proof of Theorem 3.2.2

A.10 Proof of Theorems 3.2.3 and 3.3.3

A.11 Proof of Theorem 3.3.1

A.12 Proof of Theorem 3.3.2

A.13 Proof of Theorem 5.3.1
A.14 Proof of Theorem 6.2.1 .................................................. 158
A.15 Survival analysis of IEC .................................................. 158
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Illustration of KCC Convergence with different utility functions.</td>
<td>20</td>
</tr>
<tr>
<td>2.2</td>
<td>Comparison of Clustering Quality with GP (GCC) or HCC.</td>
<td>22</td>
</tr>
<tr>
<td>2.3</td>
<td>Impact of the Number of Basic partitions to KCC.</td>
<td>23</td>
</tr>
<tr>
<td>2.4</td>
<td>Distribution of Clustering Quality of Basic partitions.</td>
<td>24</td>
</tr>
<tr>
<td>2.5</td>
<td>Sorted Pair-wise Similarity Matrix of Basic partitions.</td>
<td>24</td>
</tr>
<tr>
<td>2.6</td>
<td>Performance of KCC Based on Stepwise Deletion Strategy.</td>
<td>25</td>
</tr>
<tr>
<td>2.7</td>
<td>Quality Improvements of KCC by Adjusting RPS.</td>
<td>27</td>
</tr>
<tr>
<td>2.8</td>
<td>Quality Improvements of Basic partitions by Adjusting RPS.</td>
<td>27</td>
</tr>
<tr>
<td>2.9</td>
<td>Improvements of KCC by Using RFS.</td>
<td>28</td>
</tr>
<tr>
<td>2.10</td>
<td>The Improvement of Basic partitions by Using RFS on wine.</td>
<td>28</td>
</tr>
<tr>
<td>2.11</td>
<td>Performances of KCC on Basic partitions with Missing Data.</td>
<td>29</td>
</tr>
<tr>
<td>3.1</td>
<td>Impact of quality and quantity of basic partitions.</td>
<td>41</td>
</tr>
<tr>
<td>3.2</td>
<td>Performance of SEC with different incompleteness ratios.</td>
<td>43</td>
</tr>
<tr>
<td>3.3</td>
<td>Clustering results of partial multi-view data.</td>
<td>47</td>
</tr>
<tr>
<td>4.1</td>
<td>Framework of IEC. We apply marginalized Denoising Auto-Encoder to generate</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>infinite ensemble members by adding drop-out noise and fuse them into the consensus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>one. The figure shows the equivalent relationship between IEC and mDAE.</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>Sample Images. (a) MNIST is a 0-9 digits data sets in grey level, (b) ORL is an object data set with 100 categories, (c) ORL contains faces of 40 people with different poses and (d) Sun09 is an object data set with different types of cars.</td>
<td>58</td>
</tr>
<tr>
<td>4.3</td>
<td>Running time of linear IEC with different layers and instances.</td>
<td>61</td>
</tr>
<tr>
<td>4.4</td>
<td>Performance of linear and non-linear IEC on 13 data sets.</td>
<td>62</td>
</tr>
<tr>
<td>4.5</td>
<td>(a) Performance of IEC with different layers. (b) Impact of basic partition generation strategies. (c) Impact of the number of basic partitions via different ensemble methods on USPS. (d) Performance of IEC with different noise levels.</td>
<td>63</td>
</tr>
<tr>
<td>4.6</td>
<td>The co-association matrices with different numbers of basic partitions on USPS.</td>
<td>64</td>
</tr>
</tbody>
</table>
4.7 Survival analysis of different clustering methods in the one-omics setting. The color represents the $-\log(p\text{-value})$ of the survival analysis. The larger value indicates the more significant difference among different subgroups according to the partition by different clustering methods. For better visualization, we set the white color to be $-\log(0.05)$ so that the warm colors mean the pass of hypothesis test and the cold colors mean the failure of hypothesis test. The detailed numbers of p-value can be found in Table A.1, A.2, A.3 and A.4 in Appendix.

4.8 Number of passed hypothesis tests of different clustering methods.

4.9 Execution time in logarithm scale of different ensemble clustering methods on 13 cancer data sets with 4 different molecular types.

4.10 Survival analysis of IEC in the pan-omics setting. The value denotes the $-\log(p\text{-value})$ of the survival analysis. The detailed numbers of p-value can be found in Table A.5 in Appendix.

4.11 Survival curves of four cancer data sets by IEC.

5.1 The comparison between pairwise constraints and partition level side information. In (a), we cannot decide a Must-Link or Cannot-link only based on two instances; compared (b) with (c), it is more natural to label the instances in well-organised way, such as partition level rather than pairwise constraint.

5.2 Impact of $\lambda$ on satimage and pendigits.

5.3 Improvement of constrained clustering on glass and wine compared with K-means.

5.4 Impact of noisy side information on breast and pendigits.

5.5 Impact of the number of side information.

5.6 Performance with inconsistent cluster number on four large scale data sets.

5.7 Illustration of the proposed SG-PLCC model.

5.8 Cosegmentation results of SG-PLCC on six image groups.

5.9 Some challenging examples for our SG-PLCC model.

6.1 Some image examples of Office+Caltech (a) and PIE (b), where they have four and five subsets (domains), respectively.

6.2 Performance (%) improvement of our algorithm in the multi-source setting compared to single source setting with SURF features. The blue and red bars denote two source domains, respectively. For example, in the first bar C,W $\rightarrow$ A, the blue bar shows the improvement of our method with two source domains C and W over the one only with the source domain C.

6.3 Parameter analysis of $\lambda$ with SURF feature on Office+Caltech.

6.4 Performance (%) improvement of our algorithm in the single source setting over K-means with deep features. The letter on each bar denote the source domain.

6.5 Convergence study of our proposed method on PIE database with 5, 29 $\rightarrow$ 9 setting.
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Sample Instances of the Point-to-Centroid Distance</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Contingency Matrix</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>Sample KCC Utility Functions</td>
<td>13</td>
</tr>
<tr>
<td>2.4</td>
<td>Adjusted Contingency Matrix</td>
<td>16</td>
</tr>
<tr>
<td>2.5</td>
<td>Some Characteristics of Real-World Data Sets</td>
<td>18</td>
</tr>
<tr>
<td>2.6</td>
<td>Comparison of Execution Time (in seconds)</td>
<td>19</td>
</tr>
<tr>
<td>2.7</td>
<td>KCC Clustering Results (by $R_n$)</td>
<td>20</td>
</tr>
<tr>
<td>3.1</td>
<td>Experimental Data Sets for Scenario I</td>
<td>38</td>
</tr>
<tr>
<td>3.2</td>
<td>Clustering Results (by $R_n$) and Running Time (by sec.) in Scenario I</td>
<td>42</td>
</tr>
<tr>
<td>3.3</td>
<td>Experimental Data Sets for Scenario II</td>
<td>44</td>
</tr>
<tr>
<td>3.4</td>
<td>Clustering Results in Scenario II (by $R_n$)</td>
<td>46</td>
</tr>
<tr>
<td>3.5</td>
<td>Clustering Results in Scenario II with pseudo views (by $R_n$)</td>
<td>46</td>
</tr>
<tr>
<td>3.6</td>
<td>Sample Weibo Clusters Characterized by Keywords</td>
<td>48</td>
</tr>
<tr>
<td>4.1</td>
<td>Experimental Data Sets</td>
<td>57</td>
</tr>
<tr>
<td>4.2</td>
<td>Clustering Performance of Different Algorithms Measured by Accuracy</td>
<td>59</td>
</tr>
<tr>
<td>4.3</td>
<td>Clustering Performance of Different Algorithms Measured by NMI</td>
<td>60</td>
</tr>
<tr>
<td>4.4</td>
<td>Execution time of different ensemble clustering methods by second</td>
<td>61</td>
</tr>
<tr>
<td>4.5</td>
<td>Some key characteristics of 13 real-world datasets from TCGA</td>
<td>65</td>
</tr>
<tr>
<td>5.1</td>
<td>Notations</td>
<td>79</td>
</tr>
<tr>
<td>5.2</td>
<td>Experimental Data Sets</td>
<td>86</td>
</tr>
<tr>
<td>5.3</td>
<td>Clustering performance on seven real datasets by NMI</td>
<td>87</td>
</tr>
<tr>
<td>5.4</td>
<td>Clustering performance on seven real datasets by $R_n$</td>
<td>88</td>
</tr>
<tr>
<td>5.5</td>
<td>Comparison of Execution Time (in seconds)</td>
<td>90</td>
</tr>
<tr>
<td>5.6</td>
<td>Clustering performance of our method and different priors on iCoseg dataset</td>
<td>96</td>
</tr>
<tr>
<td>5.7</td>
<td>Comparison of segmentation accuracy on iCoseg dataset</td>
<td>96</td>
</tr>
<tr>
<td>6.1</td>
<td>Notations</td>
<td>105</td>
</tr>
<tr>
<td>6.2</td>
<td>Performance (%) comparison on three multiple sources domain benchmarks using SURF features</td>
<td>115</td>
</tr>
<tr>
<td>6.3</td>
<td>Performance (%) comparison on Office+Caltech with one source using SURF features</td>
<td>116</td>
</tr>
<tr>
<td>6.4</td>
<td>Performance (%) of our algorithm on Office+Caltech of our method with two source domains using SURF features</td>
<td>117</td>
</tr>
</tbody>
</table>
6.5 Performance (%) on Office+Caltech with one source domain using deep features or deep models ................................................................. 119
6.6 Performance (%) comparison on Office+Caltech with multi-source domains using deep features ..................................................... 120
6.7 Performance (%) on PIE with one or multi-source and one target setting ................................................................. 123

A.1 Survival analysis of different clustering algorithms on protein expression data ................................................................. 158
A.2 Survival analysis of different clustering algorithms on miRNA expression data ................................................................. 159
A.3 Survival analysis of different clustering algorithms on mRNA expression data ................................................................. 159
A.4 Survival analysis of different clustering algorithms on SCNA data ................................................................. 160
A.5 Survival analysis of IEC on pan-omics gene expression ................................................................. 160
Acknowledgments

I would like to express my deepest and sincerest gratitude to my advisor, Prof. Yun Raymond Fu, for his continuous guidance, advice, effort, patience, and encouragement during the past four years. The strong supports from Prof. Fu lie in academic and daily aspects, even in job searching. When I was in need, he is always willing to provide the help. I am truly fortunate to have him as my advisor. This dissertation and my current achievements would not have been possible without his tremendous help.

I would also like to thank my committee members, Prof. Jennifer Dy and Lu Wang for their valuable time, insightful comments and suggestions ever since my PhD research proposal. I am honored to have an opportunity to work with Prof. Dy and her student to accomplish a great paper.

I would like to thank Prof. Jiawei Han, Prof. Hui Xiong and Prof. Yu-Yang Liu for strong supports on my faculty job searching.

In addition, I would like to thank all the members from SMILE Lab, especially my coauthors and collaborators, Prof. Junjie Wu, Prof. Dacheng Tao, Prof. Tongliang Liu, Prof. Ming Shao, Dr. Jun Li, Dr. Sheng Li, Handong Zhao, Zhengming Ding, Zhiqiang Tao, Yue Wu, Kai Li, Yunlun Zhang, Junxiang Chen. For other lab members, Prof. Zhao, Dr. Yu Kong, Kang Li, Joe, Kunpeng Li, Songyao Jiang, Shuhui Jiang, Lichen Wang, Shuyang Wang, Bin Sun, Haiyi Mao, I also thank you very much. I have spent my wonderful four years with these excellent colleagues and left the impressive memory.

I would like to express my gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this dissertation. I also want to thank my fiancee, Dr. Xue Li, who came to Boston and accompanied me for one year. This dissertation and the complete PhD program would not have been possible without my family love and supports.
Abstract of the Dissertation

K-means-based Consensus Clustering: Algorithms, Theory and Applications

by

Hongfu Liu

Doctor of Philosophy in Electrical and Computer Engineering
Northeastern University, 2018
Dr. Yun Fu, Advisor

Consensus clustering aims to find a single partition which agrees as much as possible with existing basic partitions, which emerges as a promising solution to find cluster structures from heterogeneous data. It has been widely recognized that consensus clustering is effective to generate robust clustering results, detect bizarre clusters, handle noise, outliers and sample variations, and integrate solutions from multiple distributed sources of data or attributes. Different from the traditional clustering methods, which directly conducts the data matrix, the input of consensus clustering is the set of various diverse basic partitions. Therefore, consensus clustering is a fusion problem in essence, rather than a traditional clustering problem. In this thesis, we aim to solve the challenging consensus clustering by transforming it into other simple problems. Generally speaking, we propose K-means-based Consensus Clustering (KCC), which exactly transforms the consensus clustering problem into a K-means clustering problem with theoretical supports, and provide the sufficient and necessary condition of KCC utility functions. Further, based on co-association matrix we propose spectral ensemble clustering, and solve it with a weighted K-means. By this means, we decrease the time and space complexities from $O(n^3)$ and $O(n^2)$ to both $O(n)$. Finally, we achieve Infinite Ensemble Clustering with a mature technique named marginalized denoising auto-encoder. Derived from consensus clustering, a partition level constraint is proposed as the new side information for constraint clustering and domain adaptation.
Chapter 1

Introduction

1.1 Background

Cluster analysis aims at separating a set of data points into several groups so that the points in the same group are more similar than those in different groups. It is a crucial and fundamental technique in machine learning and data mining, which has been widely used in information retrieval, recommendation systems, biological analysis, and many more. A lot of efforts have been devoted to this research area, and many clustering algorithms have been proposed based on different assumptions. For example, K-means is the archetypal clustering method, which aims at finding $K$ centers to represent the whole data; Agglomerative Hierarchy Clustering merges the nearest two points or clusters at each time until all the points are in the same cluster; DBSCAN separates the points by high density regions. Since cluster analysis is an unsupervised task and different algorithms provide different clustering results, it is difficult to choose the best algorithm for a given application. Moreover, some algorithms have many parameters to tune and their performance is prone to large volatility.

Consensus clustering, also known as ensemble clustering, has been proposed as a robust meta-clustering algorithm. The algorithm fuses several diverse clustering results into an integrated one. It has been widely recognized that consensus clustering can help to generate robust clustering results, detect bizarre clusters, handle noise, outliers and sample variations, and integrate solutions from multiple distributed sources of data or attributes. Consensus clustering is a fusion problem in essence, rather than a traditional clustering problem. Consensus clustering can be generally divided into two categories. The first category designs a utility function, which measures the similarity between basic partitions and the final one, and solves a combinatorial optimization problem by
CHAPTER 1. INTRODUCTION

maximizing the utility function. The second category employs a co-association matrix to calculate
how many times a pair of instances occur in the same cluster, and then runs some graph partition
method for the final consensus result.

In this thesis, we focus on the consensus clustering, both the utility function and co-
association matrix based methods. By deep insights, we transform the challenging consensus
clustering methods into simple K-means or weighted K-means. Inspired by the consensus clustering,
especially on the utility function, the structure-preserved learning framework is designed and applied
in constraint clustering and domain adaptation. our major contributions lie in building connections
between different domains, and transforming complex problems into simple ones.

1.2 Related Work

Ensemble clustering aims to fuse various existing basic partitions into a consensus one,
which can be divided into two categories: with or without explicit global objective functions. In a
global objective function, usually a utility function is employed to measure the similarity between a
basic partition and the consensus one at the partition level. Then the consensus partition is achieved
by maximizing the summarized utility function. In the inspiring work, Ref. [7] proposed a Quadratic
Mutual Information based objective function for consensus clustering, and used K-means clustering to
find the solution. Further, they used the expectation-maximization algorithm with a finite mixture of
multinomial distributions for consensus clustering [30]. Wu et al. put forward a theoretic framework
for K-means-based Consensus Clustering (KCC), and gave the sufficient and necessary condition for
KCC utility functions that can be maximized via a K-means-like iterative process [49, 46, 50, 51].
In addition, there are some other interesting objective functions for consensus clustering, such
as the ones based on nonnegative matrix factorization [9], kernel-based methods [31], simulated
annealing [10], etc.

Another kind of methods do not set explicit global objective functions for consensus clus-
tering. In one pioneer work, Ref. [1] (GCC) developed three graph-based algorithms for consensus
clustering. More methods, however, employ co-association matrix to calculate how many times
two instances jointly belong to the same cluster. By this means, some traditional graph partitioning
methods can be called to find the consensus partition. Ref. [6] (HCC) is the most representative one in
the link-based methods, which applied the agglomerative hierarchical clustering on the co-association
matrix to find the consensus partition. Huang et al. employed the micro-cluster concept to summarize
the basic partitions into a small core co-association matrix, and applied different partitioning methods,
such as probability trajectory accumulation (PTA) and probability trajectory based graph partitioning (PTGP) \cite{52}, and graph partitioning with multi-granularity link analysis (MGLA) \cite{54}, for the final partition. Other methods include Relabeling and Voting \cite{19}, Robust Evidence Accumulation with weighting mechanism \cite{54}, Locally Adaptive Cluster based methods \cite{20}, Robust Spectral Ensemble Clustering \cite{55} and Simultaneous Clustering and Ensemble \cite{56}, etc. There are still many other algorithms for ensemble clustering. Readers with interests can refer to some survey papers for more comprehensive understanding \cite{11}. Most of the existing works focus on the process of the clustering on the (modified) co-association matrix.

1.3 Dissertation Organization

The rest of this dissertation is organized as follows.

Chapter 2 introduces the K-means-based Consensus Clustering (KCC), where we propose KCC utility functions and link it to flexible divergences. With this method, a rich family of KCC utility functions in consensus clustering can be efficiently solved by K-means on a binary matrix with theoretical supports.

In Chapter 3, Spectral Ensemble Clustering (SEC) is put forward, which applies the spectral clustering on the co-association matrix. To solve SEC efficiently, the co-association graph is decomposed into the binary matrix, where weighted K-means is conducted for the final solution. This method dramatically decreases the time and space complexities from $O(n^3)$ and $O(n^2)$ to both roughly $O(n)$.

Chapter 4 delivers the Infinite Ensemble Clustering (IEC), which aims to fuse infinite basic partitions for robust solution. To achieve this, we build the equivalent connection between IEC and marginalized denoising auto-encoder. By this means, IEC can be dealt with a mature technique in a closed-form solution.

Inspired by consensus clustering, the utility function is employed to measure the similarity in the partition-level. Chapter 5 and Chapter 6 are two applications in terms of constraint clustering and domain adaptation. Generally speaking, we use the utility function to preserve the structure of side information or source data for target data exploration.

Finally, Chapter 7 concludes this dissertation.
Chapter 2

K-means-based Consensus Clustering

Consensus clustering, also known as cluster ensemble or clustering aggregation, aims to find a single partition of data from multiple existing basic partitions \([1][2]\). It has been widely recognized that consensus clustering can be helpful for generating robust clustering results, finding bizarre clusters, handling noise, outliers and sample variations, and integrating solutions from multiple distributed sources of data or attributes \([3]\).

In the literature, many algorithms have been proposed to address the computational challenges, such as the co-association matrix based methods \([6]\), the graph-based methods \([1]\), the prototype-based methods \([7]\), and other heuristic approaches \([8][9][10]\). Among these research efforts, the K-means-based method proposed in Ref. \([7]\) is of particular interests, for its simplicity and high efficiency inherited from classic K-means clustering methods. However, the existing studies along this line are still preliminary and fragmented. Indeed, the general theoretic framework of utility functions suitable for K-means-based consensus clustering (KCC) is yet not available. Also, the understanding of key factors, which have significant impact on the performances of KCC, is still limited.

To fulfill this crucial void, in this chapter, we provide a systematic study of K-means-based consensus clustering. The major contributions are summarized as follows. First, we formally define the concept of KCC, and provide a necessary and sufficient condition for utility functions which are suitable for KCC. Based on this condition, we can easily derive a KCC utility function from a continuously differentiable convex function, which helps to establish a unified framework for KCC, and makes it a systematic solution. Second, we redesign the computation procedures of utility functions and distance functions for KCC. This redesign helps to successfully extend the applicable scope of KCC to the cases where there exist severe data incompleteness. Third, we empirically
explore the major factors which can affect the performances of KCC, and obtain some practical
guidance from specially designed experiments on various real-world data sets.

Extensive experiments on various real-world data sets demonstrate that: (a) KCC is highly
efficient and is comparable to the state-of-the-art methods in terms of clustering quality; (b) Multiple
utility functions indeed improve the usability of KCC on different types of data, while we find that
the utility functions based on Shannon entropy generally have more robust performances; (c) KCC is
very robust even if there exist very few high-quality basic partitions or severely incomplete basic
partitions; (d) The choice of the generation strategy for basic partitions is critical to the success of
KCC; (e) The number, quality and diversity of basic partitions are three major factors that affect the
performances of KCC, while the impacts from them are different.

2.1 Preliminaries and Problem Definition

In this section, we briefly introduce the basic concepts of consensus clustering and K-means
clustering, and then formulate the problem to be studied in this chapter.

2.1.1 Consensus Clustering

We begin by introducing some basic mathematical notations. Let \( X = \{x_1, x_2, \cdots, x_n\} \)
denote a set of data objects/points/instances. A partition of \( X \) into \( K \) crisp clusters can be represented
as a collection of \( K \) subsets of objects in \( C = \{C_k | k = 1, \cdots, K\} \), with \( C_k \cap C_{k'} = \emptyset \), \( \forall k \neq k' \),
and \( \bigcup_{k=1}^{K} C_k = X \), or as a label vector \( \pi = (L_{\pi}(x_1), \cdots, L_{\pi}(x_n)) \), where \( L_{\pi}(x_i) \) maps \( x_i \) to one
of the \( K \) labels in \( \{1, 2, \cdots, K\} \). We also use some conventional mathematical notations as follows.
For instance, \( \mathbb{R}, \mathbb{R}_+, \mathbb{R}_{++}, \mathbb{R}^d \) and \( \mathbb{R}^{nd} \) are used to denote the sets of reals, non-negative reals,
positive reals, \( d \)-dimensional real vectors, and \( n \times d \) real matrices, respectively. \( \mathbb{Z} \) denotes the set
of integers, and \( \mathbb{Z}_+, \mathbb{Z}_{++}, \mathbb{Z}^d \) and \( \mathbb{Z}^{nd} \) are defined analogously. For a \( d \)-dimensional real vector
\( x \), \( \|x\|_p \) denotes the \( L_p \) norm of \( x \), i.e., \( \|x\|_p = \sqrt[p]{\sum_{i=1}^{d} x_i^p} \), \( |x| \) denotes the cardinality of \( x \), i.e.,
\( |x| = \sum_{i=1}^{d} x_i \), and \( x^T \) denotes the transposition of \( x \). The gradient of a single variable function \( f \) is
denoted as \( \nabla f \), and the logarithm of based 2 is denoted as \( \log \).

In general, the existing consensus clustering methods can be categorized into two classes,
i.e., the methods with and without global objective functions, respectively \( [\Pi] \). In this chapter, we are
concerned with the former methods, which are typically formulated as a combinatorial optimization
problem as follows. Given \( r \) basic crisp partitions of \( X \) \( (a \text{ basic partition is a partition of } \mathcal{X} \text{ given by} \)
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

running some clustering algorithm on $X$ in $\Pi = \{\pi_1, \pi_2, \cdots, \pi_r\}$, the goal is to find a consensus partition $\pi$ such that

$$\Gamma(\pi, \Pi) = \sum_{i=1}^{r} w_i U(\pi, \pi_i)$$

(2.1)

is maximized, where $\Gamma : \mathbb{Z}_{n+}^n \times \mathbb{Z}_{r+}^n \mapsto \mathbb{R}$ is a consensus function, $U : \mathbb{Z}_{n+}^n \times \mathbb{Z}_{n+}^n \mapsto \mathbb{R}$ is a utility function, and $w_i \in [0, 1]$ is a user-specified weight for $\pi_i$, with $\sum_{i=1}^{r} w_i = 1$. Sometimes a distance function, e.g., the well-known Mirkin distance $[23]$, rather than a utility function is used in the consensus function. In that case, we can simply turn the maximization problem into a minimization problem without changing the nature of the problem.

It has been proven that consensus clustering is an NP-complete problem, which implies that it can be solved only by some heuristics and/or some meta-heuristics. Therefore, the choice of the utility function in Eq. (2.1) is crucial for the success of a consensus clustering, since it largely determines the heuristics to employ. In the literature, some external measures originally proposed for cluster validity have been adopted as the utility functions for consensus clustering, such as the Normalized Mutual Information $[11]$, Category Utility Function $[29]$, Quadratic Mutual Information $[7]$, and Rand Index $[10]$. These utility functions, usually possessing different mathematical properties, pose computational challenges to consensus clustering.

2.1.2 K-means Clustering

K-means $[35]$ is a prototype-based, simple partitional clustering technique, which attempts to find user-specified $K$ crisp clusters. These clusters are represented by their centroids — usually the arithmetic means of data points in the respective clusters. K-means can be also viewed as a heuristic to optimize the following objective function:

$$\min_{k=1}^{K} \sum_{x \in C_k} f(x, m_k),$$

(2.2)

where $m_k$ is the centroid of the $k$th cluster $C_k$, and $f$ is the distance function$^1$ that measures the distance from a data point to a centroid.

The clustering process of K-means is a two-phase iterative heuristic as follows. First, $K$ initial centroids are selected, where $K$ is the desired number of clusters specified by the users. Every point in the data set is then assigned to the closest centroid in the assigning phase, and each collection of points assigned to a centroid forms a cluster. The centroid of each cluster is then updated in the

$^1$Here the K-means distance function is a general concept, which might not hold the properties of a distance function.
**CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING**

Table 2.1: Sample Instances of the Point-to-Centroid Distance

<table>
<thead>
<tr>
<th>φ(x)</th>
<th>dom(φ)</th>
<th>f(x, y)</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>∥x∥²</td>
<td>Rd</td>
<td>∥x − y∥²</td>
<td>squared Euclidean distance</td>
</tr>
<tr>
<td>−H(x)</td>
<td>{x</td>
<td>x ∈ Rd⁺,</td>
<td>x</td>
</tr>
<tr>
<td>∥x∥₂</td>
<td>Rd⁺⁺</td>
<td>∥x∥₂(1 − cos(x, y))</td>
<td>cosine distance</td>
</tr>
<tr>
<td>∥x∥ₚ, p &gt; 1</td>
<td>Rd⁺⁺</td>
<td>φ(x) − ∑₁≤j≤d xj/yj⁻¹</td>
<td>L_p distance</td>
</tr>
</tbody>
</table>

Note: (1) H: Shannon entropy; (2) cos: cosine similarity.

*updating* phase, based on the points assigned to that cluster. This process is repeated until no points change clusters or some stopping criteria are met. This process is also known as the *centroid-based alternating optimization* method from an optimization perspective [36], which has some well-known advantages compared with a wide range of other methods, such as simplicity, high efficiency, and satisfactory accuracy.

It is interesting to note that the choice of distance functions in Eq. (2.2) is closely related to the choice of centroid types in K-means, given that the convergence of the two-phase iteration must be guaranteed [37]. For instance, if the well-known squared Euclidean distance is used, the centroids must be the arithmetic means of cluster members. However, if the city-block is used instead, the centroids must be the medians. Since the arithmetic mean has higher computational efficiency and better analytical properties, we hereby limit our study to the classic K-means with arithmetic centroids. We call a distance function *fits* K-means if this function corresponds to the centroids of arithmetic means.

It has been shown that the Bregman divergence [38] fits the classic K-means as a family of distances [39]. In other words, let φ : Rd ↦ Rd be a differentiable, strictly-convex function, then the Bregman loss function f : Rd × Rd ↦ Rd defined by

$$f(x, y) = \phi(x) − \phi(y) − (x − y)ᵀ∇\phi(y)$$

(2.3)

fits K-means clustering. More importantly, under some assumptions including the *unique minimizer* assumption on the centroids, the Bregman divergence is the only distance that fits K-means [40]. Nevertheless, the strictness of the convexity of φ can be further relaxed, if the unique minimizer assumption reduces to the non-unique case. This leads to the more general “point-to-centroid distance” derived from convex but not necessarily strictly convex φ [41], although it has the same mathematical expression as the Bregman divergence in Eq. (2.3). As a result, we can reasonably assume that the distance function f in Eq. (2.2) is an instance of the point-to-centroid distance. Table 2.1 lists some important instances of the point-to-centroid distance.
2.1.3 Problem Definition

In general, there are three key issues for a consensus clustering algorithm: accuracy, efficiency, and flexibility. Accuracy means the algorithm should be able to find a high-quality partition from a large combinatorial space. The simple heuristics, such as the Best-of-\(r\) algorithm that only selects the best partition from the \(r\) basic partitions, usually cannot guarantee satisfactory results. Efficiency is another big challenge to consensus clustering, especially for the algorithms that employ some complicated meta-heuristics, such as the genetic algorithm, simulated annealing, and the particle swarm algorithm. Flexibility requires the algorithm to serve as a framework open to various types of utility functions. This is important for the applicability of the algorithm to a wide range of application domains. However, the flexibility issue has seldom been addressed in the literature, since most of the existing algorithms either have no objective functions, or are designed purposefully for one specific utility function.

Then, here is the problem: Can we design a consensus clustering algorithm that can address the three problems simultaneously? The forementioned K-means algorithm indeed provides an interesting clue. If we can somehow transform the consensus clustering problem into a K-means clustering problem, we can then make use of the two-phase iteration heuristic of K-means to find a good consensus partition in an efficient way. Moreover, as indicated by Eq. 2.3, the point-to-centroid distance of K-means is actually a family of distance functions derived from different convex functions \(\phi\). This implies that if the distance function of K-means can be mapped to the utility function of consensus clustering, we can obtain multiple utility functions for consensus clustering in a unified framework, which will provide great flexibility.

In light of this, in this chapter, we focus on building a general framework for consensus clustering using K-means, which is referred to as the K-means-based Consensus Clustering (KCC) method. We are concerned with the following three questions:

- How to transform a consensus clustering problem to a K-means clustering problem?
- What is the necessary and sufficient condition for this transformation?
- How to adapt KCC to the situations where there exist incomplete basic partitions?

Here “incomplete basic partition” means a basic partition that misses some of the data labels. It may due to the unavailability of some data objects in a distributed or time-evolving system, or simply the loss of some data labels in a knowledge reuse process [1].
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

Table 2.2: Contingency Matrix

<table>
<thead>
<tr>
<th></th>
<th>$C_1^{(i)}$</th>
<th>$C_2^{(i)}$</th>
<th>$\cdots$</th>
<th>$C_{K_i}^{(i)}$</th>
<th>$\sum$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi$</td>
<td>$n_{11}^{(i)}$</td>
<td>$n_{12}^{(i)}$</td>
<td>$\cdots$</td>
<td>$n_{1K_i}^{(i)}$</td>
<td>$n_{1+}$</td>
</tr>
<tr>
<td></td>
<td>$n_{21}^{(i)}$</td>
<td>$n_{22}^{(i)}$</td>
<td>$\cdots$</td>
<td>$n_{2K_i}^{(i)}$</td>
<td>$n_{2+}$</td>
</tr>
<tr>
<td></td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
</tr>
<tr>
<td></td>
<td>$n_{K1}^{(i)}$</td>
<td>$n_{K2}^{(i)}$</td>
<td>$\cdots$</td>
<td>$n_{KK_i}^{(i)}$</td>
<td>$n_{K+}$</td>
</tr>
<tr>
<td>$\sum$</td>
<td>$n_{+1}^{(i)}$</td>
<td>$n_{+2}^{(i)}$</td>
<td>$\cdots$</td>
<td>$n_{+K_i}^{(i)}$</td>
<td>$n$</td>
</tr>
</tbody>
</table>

Note that few studies have investigated consensus clustering from a KCC perspective except for Ref. [7], which demonstrated that using the special Category Utility Function, a consensus clustering can be equivalently transformed to a K-means clustering with the squared Euclidean distance [29]. However, it did not establish a unified framework for KCC, neither did it explore the general property of utility functions that fit KCC. Moreover, it did not showcase how to handle incomplete basic partitions, which as shown in a later section is indeed a big challenge to KCC. We therefore attempt to fill these voids and make KCC as one representative solution for consensus clustering in practice.

2.2 Utility Functions for K-means-based Consensus Clustering

In this section, we first establish the general framework of K-means-based consensus clustering (KCC). We then propose the necessary and sufficient condition for a utility function suitable for KCC (referred to as a KCC utility function), and show how to link it to the K-means clustering. We finally highlight two special forms of KCC utility functions for practical purpose.

2.2.1 From Consensus Clustering to K-means Clustering

We begin by introducing the notion of contingency matrix. A contingency matrix is actually a co-occurrence matrix for two discrete random variables. It is often used for computing the difference or similarity between two partitions in cluster validity. Table 2.2 shows a typical example.

In Table 2.2 we have two partitions: $\pi$ and $\pi_i$, containing $K$ and $K_i$ clusters, respectively. In the table, $n_{kj}^{(i)}$ denotes the number of data objects belonging to both cluster $C_j^{(i)}$ in $\pi_i$ and cluster $C_k$ in $\pi$, $n_{k+} = \sum_{j=1}^{K_i} n_{kj}^{(i)}$, and $n_{+j}^{(i)} = \sum_{k=1}^{K} n_{kj}^{(i)}$, $1 \leq j \leq K_i$, $1 \leq k \leq K$. Let $p_{kj}^{(i)} = n_{kj}^{(i)}/n$, $p_{k+} = n_{k+}/n$, and $p_{+j}^{(i)} = n_{+j}^{(i)}/n$, we then have the normalized contingency matrix (NCM), based on which a wide range of utility functions can be defined accordingly. For instance, the well-known
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

Category Utility Function \cite{29} can be computed upon NCM as follows:

\[
U_c(\pi, \pi_i) = \sum_{k=1}^{K} p_k + \sum_{j=1}^{K_i} \left( \frac{p_{kj}}{p_{k+}} \right)^2 - \sum_{j=1}^{K_i} \left( \frac{p_{ij}}{p_{j+}} \right)^2.
\]  

(2.4)

We then introduce the notion of binary data. Let \(X^{(b)} = \{x_l^{(b)} | 1 \leq l \leq n\}\) be a binary data set derived from the set of \(r\) basic partitions \(\Pi\) as follows:

\[
x_l^{(b)} = \langle x_{l,1}^{(b)}, \ldots, x_{l,i}^{(b)}, \ldots, x_{l,r}^{(b)} \rangle, \quad\text{with}\quad (2.5)
\]

\[
x_{l,i}^{(b)} = \langle x_{l,i,1}, \ldots, x_{l,i,j}, \ldots, x_{l,i,K_i} \rangle, \quad\text{and}\quad (2.6)
\]

\[
x_{l,i,j}^{(b)} = \begin{cases} 
1, & \text{if } L_{\pi_i}(x_l) = j \\
0, & \text{otherwise}
\end{cases}, \quad (2.7)
\]

where “\(\langle \rangle\)" indicates a transversal vector. Therefore, \(X^{(b)}\) is an \(n \times \sum_{i=1}^{r} K_i\) binary data matrix with \(|x_{l,i}^{(b)}| = 1, \forall l, i\).

Now suppose we have a partition \(\pi\) with \(K\) clusters generated by running K-means on \(X^{(b)}\). Let \(m_k\) denote the centroid of the \(k\)th cluster in \(\pi\), which is a \(\sum_{i=1}^{r} K_i\)-dimensional vector as follows:

\[
m_k = \langle m_{k,1}, \ldots, m_{k,i}, \ldots, m_{k,r} \rangle, \quad\text{with}\quad (2.8)
\]

\[
m_{k,i} = \langle m_{k,i,1}, \ldots, m_{k,i,j}, \ldots, m_{k,i,K_i} \rangle, \quad (2.9)
\]

\(1 \leq j \leq K_i, 1 \leq i \leq r, \text{ and } 1 \leq k \leq K\). We then link the binary data to the contingency matrix by formalize a lemma as follows:

**Lemma 2.2.1** For K-means clustering on the binary data set \(X^{(b)}\), the centroids satisfy

\[
m_{k,i} = \left\langle \frac{p_{k1}^{(i)}}{p_{k+}}, \ldots, \frac{p_{kj}^{(i)}}{p_{k+}}, \ldots, \frac{p_{kK_i}^{(i)}}{p_{k+}} \right\rangle, \quad \forall k, i. \quad (2.10)
\]

**Remark 1:** While Lemma 2.2.1 is very simple, it unveils important information critical to the construction of KCC framework. That is, using the binary data set \(X^{(b)}\) as the input for K-means clustering, the resulting centroids can be computed upon the elements in the contingency matrices, from which a consensus function can be also defined. In other words, the contingency matrix and the binary data set together serve as a bridge that removes the boundary between consensus clustering and K-means clustering.

We then give the definition of KCC utility function as follows:
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

**Definition 1** A utility function $U$ is a KCC utility function, if $\forall \Pi = \{\pi_1, \cdots, \pi_r\}$ and $K \geq 2$, there exists a distance function $f$ such that

$$
\max_{\pi \in F} \sum_{i=1}^{r} w_i U(\pi, \pi_i) \Leftrightarrow \min_{\pi \in F} \sum_{k=1}^{K} \sum_{x_l \in C_k} f(x_l^{(b)}, m_k) 
$$

(2.11)

holds for any feasible region $F$.

Remark. Definition 1 specifies the key property of a KCC utility function; that is, it should help to transform a consensus clustering problem to a K-means clustering problem. In other words, KCC is a solution to consensus clustering, which takes a KCC utility function to define the consensus function, and relies on the K-means heuristic to find the consensus partition.

### 2.2.2 The Derivation of KCC Utility Functions

Here we derive the KCC utility functions, and give some examples that can be commonly used in real-world applications. We first give the following lemma:

**Lemma 2.2.2** A utility function $U$ is a KCC utility function, if and only if $\forall \Pi$ and $K \geq 2$, there exist a differentiable convex function $\phi$ and a strictly increasing function $g_{\Pi,K}$ such that

$$
\sum_{i=1}^{r} w_i U(\pi, \pi_i) = g_{\Pi,K}(\sum_{k=1}^{K} p_{k+}\phi(m_k)).
$$

(2.12)

Remark 2. Compared with the definition of the KCC utility function in Definition 1, the greatest value of Lemma 2.2.2 is to replace “$\Leftrightarrow$” by “$=$”, which sheds light on deriving the detailed expression of KCC utility functions. Note that we use $\Pi$ and $K$ as the subscripts for $g$, because these two parameters directly affect the ranking of $\pi$ in $F^*$ given by $\Upsilon$ or $\Psi$. In other words, different mapping functions may exist for different settings of $\Pi$ and $K$.

Next, we go a further step to analyze Eq. (2.12). Recall the contingency matrix in Table 2.2. Let $P^{(i)}_k$ denote $(p^{(i)}_{k1}/p_{k+}, \cdots, p^{(i)}_{kj}/p_{k+}, \cdots, p^{(i)}_{kK}/p_{k+})$ for simplicity. According to Lemma 2.2.1, $P^{(i)}_k \overset{1}{=} m_{k,i}$, but $P^{(i)}_k$ is defined more from a contingency matrix perspective. We then have the following important theorem:

**Theorem 2.2.1** $U$ is a KCC utility function, if and only if $\forall \Pi = \{\pi_1, \cdots, \pi_r\}$ and $K \geq 2$, there exists a set of continuously differentiable convex functions $\{\mu_1, \cdots, \mu_r\}$ such that

$$
U(\pi, \pi_i) = \sum_{k=1}^{K} p_{k+}\mu_i(P^{(i)}_k), \forall i.
$$

(2.13)
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

The convex function \( \Phi \) for the corresponding K-means clustering is given by

\[
\Phi(m_k) = \sum_{i=1}^{r} w_i \nu_i(m_{k,i}), \forall \ k, \tag{2.14}
\]

where

\[
\nu_i(x) = a \mu_i(x) + c_i, \forall \ i, a \in \mathbb{R}^+, c_i \in \mathbb{R}. \tag{2.15}
\]

Remark 3. Theorem 2.2.1 gives the necessary and sufficient condition for being a KCC utility function. That is, a KCC utility function must be a weighted average of a set of convex functions defined on \( P_k^{(i)} \), \( 1 \leq i \leq r \), respectively. From this perspective, Theorem 2.2.1 can serve as the criterion to verify whether a given utility function is a KCC utility function. Nevertheless, the most important thing is, Theorem 2.2.1 indicates the way to conduct the K-means-based consensus clustering. That is, we first design or designate a set of convex functions \( \mu_i \) defined on \( P_k^{(i)} \), \( 1 \leq i \leq r \), from which the utility function as well as the consensus function can be derived by Eq. (2.13) and Eq. (2.1), respectively; then after setting \( a \) and \( c_i \), \( 1 \leq i \leq r \), in Eq. (2.15), we can determine the corresponding \( \Phi \) for K-means clustering by Eq. (2.14), which is further used to derive the point-to-centroid distance \( f \) using Eq. (2.3); the K-means clustering is finally employed to find the consensus partition \( \pi \).

Remark 4. Some practical points regarding to \( \mu_i \) and \( \nu_i \), \( 1 \leq i \leq r \), are noteworthy here. First, according to Eq. (2.3), different settings of \( a \) and \( c_i \) in Eq. (2.15) will lead to different distances \( f \) but the same K-means clustering in Eq. (2.2), given that \( \mu_i \), \( 1 \leq i \leq r \), are invariant. As a result, we can simply let \( \nu_i \equiv \mu_i \) by having \( a = 1 \) and \( c_i = 0 \) in practice, which are actually the default settings in our work. Second, it is more convenient to unify \( \mu_i \), \( 1 \leq i \leq r \), to a same convex function \( \mu \) in practice, although they are treated separately above to keep the generality of the theorem. This also becomes the default setting in our work. Third, it is easy to show that the linear extension of \( \mu \) to \( \mu'(x) = c \mu(x) + d \) (\( c \in \mathbb{R}^+, d \in \mathbb{R} \)) will change the utility function in Eq. (2.13) proportionally but again leads to the same K-means clustering and thus the same consensus partition. Therefore, there is a many-to-one correspondence from utility function to K-means clustering, and we can use the simplest form of \( \mu \) without loss of accuracy.

Example. Hereinafter, we denote the KCC utility function derived by \( \mu \) in Eq. (2.13) as \( U_\mu \) for the convenience of description. Table 2.3 shows some examples of KCC utility functions derived from various convex functions \( \mu \), and their corresponding point-to-centroid distances \( f \), where \( P^{(i)} = (p_{i+1}^{(i)}, \ldots, p_{i+j}^{(i)}, \cdots, p_{i+K_i}^{(i)}) \), \( 1 \leq i \leq r \). Note that \( U_c \) is the well-known Category Utility Function \([29]\), but the other three \( U_\mu \) have hardly been mentioned in the literature. In fact, we
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

Table 2.3: Sample KCC Utility Functions

<table>
<thead>
<tr>
<th>µ(mk,i)</th>
<th>Uµ(π, πi)</th>
<th>f(xb,l,mk)</th>
</tr>
</thead>
<tbody>
<tr>
<td>uc</td>
<td>∑k=1p k∥Pk∥2 − ∥Pk∥2</td>
<td>∑i=1wi∥x(b)i∥2 − ∥mk,i∥2</td>
</tr>
<tr>
<td>uh</td>
<td>∑k=1p k∥Pk∥2 − ∥Pk∥2</td>
<td>∑i=1wiD(x(b)i∥mk,i)</td>
</tr>
<tr>
<td>ucos</td>
<td>∑k=1p k∥Pk∥2 − ∥Pk∥2</td>
<td>∑i=1wi(1 − cos(x(b)i∥mk,i))</td>
</tr>
<tr>
<td>ulp</td>
<td>∑k=1p k∥Pk∥2 − ∥Pk∥2</td>
<td>∑i=1wi(1 − ∑j=1Kk−1 x(b)j</td>
</tr>
</tbody>
</table>

Note: D means KL-divergence.

will demonstrate in the experimental section that Uc often performs the worst among these utility functions. This, in turn, justifies the necessity of providing different KCC utility functions for K-means-based consensus clustering. Moreover, it is worth noting that a constant based on P(i) is added to µ for each Uµ in Table 2.3, although it does not affect the corresponding distance function f. By adding this constant, the derived Uµ actually has an interesting physical meaning: utility gain. We will detail this in Section 2.2.3 Finally, it is also interesting to point out that the derived distance function f is just the weighted sum of the distances related to the different basic partitions. This indeed broadens the traditional scope of the distance functions that fit K-means clustering. In particular, it sheds light on employing KCC for handling inconsistent data in Section 2.3 below.

2.2.3 Two Forms of KCC Utility Functions

Theorem 2.2.1 indicates how to derive a KCC utility function from a convex function µ, or vice versa. However, it does not guarantee that the obtained KCC utility function is explainable. Therefore, we here introduce two special forms of KCC utility functions which are meaningful to some extent.

2.2.3.1 Standard Form of KCC Utility Functions

Suppose we have a utility function Uµ derived from µ. Recall P(i) = {pi+1,...,pi+Ki}, 1 ≤ i ≤ r, which are actually constant vectors given the basic partitions Π. If we let

\[ \mu_s(P_k^{(i)}) = \mu(P_k^{(i)}) - \mu(P^{(i)}) \]

then by Eq. (2.13), we can obtain a new utility function as follows:

\[ U_{\mu_s}(\pi, \pi_i) = U_{\mu}(\pi, \pi_i) - \mu(P^{(i)}) \]
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

As \( \mu(P^{(i)}) \) is a constant given \( \pi_i \), \( \mu_s \) and \( \mu \) will lead to a same corresponding point-to-centroid distance \( f \), and thus a same consensus partition \( \pi \). The advantage of using \( \mu_s \) rather than \( \mu \) roots in the following proposition:

**Proposition 2.2.1** \( U_{\mu_s} \geq 0 \).

Proposition 2.2.1 ensures the non-negativity of \( U_{\mu_s} \). Indeed, \( U_{\mu_s} \) can be viewed as the *utility gain* from a consensus clustering, by calibrating \( U_\mu \) to the benchmark: \( \mu(P^{(i)}) \). Here, we define the utility gain as the *standard form* of a KCC utility function. Accordingly, all the utility functions listed in Table 2.3 are in the standard form. It is also noteworthy that the standard form has invariance; that is, if we let \( \mu_{ss}(P_k^{(i)}) = \mu_s(P_k^{(i)}) - \mu_s(P^{(i)}) \), we have \( \mu_{ss} \equiv \mu_s \) and \( U_{\mu_{ss}} \equiv U_{\mu_s} \).

Therefore, given a convex function \( \mu \), it can derive one and only one KCC utility function in the standard form.

2.2.3.2 Normalized Form of KCC Utility Functions

It is natural to take a further step from the standard form \( U_{\mu_s} \) to the *normalized form* \( U_{\mu_n} \). Let

\[
\mu_n(P_k^{(i)}) = \frac{\mu_s(P_k^{(i)})}{|\mu(P^{(i)})|} = \frac{\mu(P_k^{(i)}) - \mu(P^{(i)})}{|\mu(P^{(i)})|},
\]

(2.18)

Since \( \mu(P^{(i)}) \) is a constant given \( \pi_i \), it is easy to know that \( \mu_n \) is also a convex function, from which a KCC utility function \( U_{\mu_n} \) can be derived as follows:

\[
U_{\mu_n}(\pi, \pi_i) = \frac{U_{\mu_s}(\pi, \pi_i)}{|\mu(P^{(i)})|} = \frac{U_\mu(\pi, \pi_i) - \mu(P^{(i)})}{|\mu(P^{(i)})|},
\]

(2.19)

From Eq. (2.19), \( U_{\mu_n} \geq 0 \), which can be viewed as the *utility gain ratio* to the constant \( |\mu(P^{(i)})| \). Note that the \( \phi \) functions corresponding to \( U_{\mu_n} \) and \( U_{\mu_s} \), respectively, are different due to the introduction of \( |\mu(P^{(i)})| \) in Eq. (2.18). As a result, the consensus partitions by KCC are also different for \( U_{\mu_n} \) and \( U_{\mu_s} \). Nevertheless, the KCC procedure for \( U_{\mu_n} \) will be exactly the same as the procedure for \( U_{\mu_s} \) or \( U_{\mu} \), if we let \( w_i = w_i / |\mu(P^{(i)})|, 1 \leq i \leq r \), in Eq. (2.14). Finally, it is easy to note that the normalized form \( U_{\mu_n} \) also has the invariance property.

In summary, given a convex function \( \mu \), we can derive a KCC utility function \( U_\mu \), as well as its standard form \( U_{\mu_s} \) and normalized form \( U_{\mu_n} \). While \( U_{\mu_s} \) leads to a same consensus partition as \( U_\mu \), \( U_{\mu_n} \) results in a different one. Given clear physical meanings, the standard form and normalized form will be adopted as two major forms of KCC utility functions in the experimental section below.
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

2.3 Handling Incomplete Basic partitions

Here, we introduce how to exploit K-means-based consensus clustering for handling incomplete basic partitions (IBPs). We begin by formulating the problem as follows.

2.3.1 Problem Description

Let $X = \{x_1, x_2, \cdots, x_n\}$ denote a set of data objects. A basic partition $\pi_i$ is obtained by clustering a data subset $X_i \subseteq X$, $1 \leq i \leq r$, with the constraint that $\bigcup_{i=1}^{r} X_i = X$. Here the problem is, given $r$ IBPs in $\Pi = \{\pi_1, \cdots, \pi_r\}$, how to cluster $\mathcal{X}$ into $K$ crisp clusters using KCC?

The value of solving this problem lies in two folds. First, from the theoretical perspective, IBPs will generate incomplete binary data set $X^{(b)}$ with missing values, and how to deal with missing values has long been the challenging problem in the statistical field. Moreover, how to guarantee the convergence of K-means on incomplete data is also very interesting in theory. Second, from the practical perspective, it is not unusual in real-world applications that part of data instances are unavailable in a basic partition due to a distributed system or the delay of data arrival. Knowledge reuse is also a source for IBPs, since the knowledge of various basic partitions may be gathered from different research or application tasks [1].

Intuitively, one can employ the traditional statistical methods to recover the missing values in an incomplete binary data set. In this way, we can still call KCC on recovered $\mathcal{X}^{(b)}$ without any modification. This method, however, is applicable only when the proportion of missing values is relatively small. The binary property of $X^{(b)}$ also limits the use of some statistics such as the mean, and some distributions such as the normal distribution.

Another solution is to add a special cluster, i.e., the missing cluster, to each basic partition. All the missing data instances in a basic partition will be assigned to the missing cluster. While this method also enables the use of KCC without any modification, it still seems weird to have a large missing cluster in a basic partition when there exists severe data incompleteness. These missing clusters actually provide no useful information to the true data structure.

To meet this challenge, in what follows, we propose a new solution to K-means-based consensus clustering on IBPs.
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

Table 2.4: Adjusted Contingency Matrix

<table>
<thead>
<tr>
<th></th>
<th>(C_1)</th>
<th>(C_2)</th>
<th>(\cdots)</th>
<th>(C_K)</th>
<th>(\sum)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\pi)</td>
<td>(n_{11}^{(i)})</td>
<td>(n_{12}^{(i)})</td>
<td>(\cdots)</td>
<td>(n_{1K}^{(i)})</td>
<td>(n_{1+}^{(i)})</td>
</tr>
<tr>
<td>(C_1)</td>
<td>(n_{21}^{(i)})</td>
<td>(n_{22}^{(i)})</td>
<td>(\cdots)</td>
<td>(n_{2K}^{(i)})</td>
<td>(n_{2+}^{(i)})</td>
</tr>
<tr>
<td>(\cdots)</td>
<td>(\cdots)</td>
<td>(\cdots)</td>
<td>(\cdots)</td>
<td>(\cdots)</td>
<td>(\cdots)</td>
</tr>
<tr>
<td>(C_K)</td>
<td>(n_{K1}^{(i)})</td>
<td>(n_{K2}^{(i)})</td>
<td>(\cdots)</td>
<td>(n_{KK}^{(i)})</td>
<td>(n_{K+}^{(i)})</td>
</tr>
<tr>
<td>(\sum)</td>
<td>(n_{+1}^{(i)})</td>
<td>(n_{+2}^{(i)})</td>
<td>(\cdots)</td>
<td>(n_{+K}^{(i)})</td>
<td>(n^{(i)})</td>
</tr>
</tbody>
</table>

2.3.2 Solution

We first adjust the way for utility computation on IBPs. We still have maximizing Eq. (2.1) as the objective of consensus clustering, but the contingency matrix for \(U(\pi, \pi_i)\) computation is modified to the one in Table 2.4. In the table, \(n_{kj}^{(i)}\) is the number of instances assigned from \(X_i\) to cluster \(C_k\), \(1 \leq k \leq K\), and \(n^{(i)}\) is the total number of instances in \(X_i\), i.e., \(n^{(i)} = |X_i|\), \(1 \leq i \leq r\). Let \(p_{kj}^{(i)} = n_{kj}^{(i)}/n^{(i)}\), \(p_{k+}^{(i)} = n_{k+}^{(i)}/n^{(i)}\), \(p_{+j}^{(i)} = n_{+j}^{(i)}/n^{(i)}\), and \(p^{(i)} = n^{(i)}/n\).

We then adjust K-means clustering to handle the incomplete binary data set \(X^{(b)}\). Let the distance \(f\) be the sum of the point-to-centroid distances in different basic partitions, i.e.,

\[
f(x^{(b)}_i, m_k) = \sum_{i=1}^{r} I(x_i \in X_i) f_i(x^{(b)}_{i,i}, m_{k,i}),
\]

where \(f_i\) is \(f\) on the \(i\)th “block” of \(X^{(b)}\). \(I(x_i \in X_i) = 1\) if \(x_i \in X_i\), and 0 otherwise.

We then obtain a new objective function for K-means clustering as follows:

\[
F = \sum_{k=1}^{K} \sum_{x_i \in C_k} f(x^{(b)}_i, m_k)
\]

\[
= \sum_{i=1}^{r} \sum_{k=1}^{K} \sum_{x_i \in C_k \cap X_i} f_i(x^{(b)}_{i,i}, m_{k,i}),
\]

where the centroid \(m_{k,i} = (m_{k,i1}, \cdots, m_{k,iK})\), with

\[
m_{k,ij} = \frac{\sum_{x_i \in C_k \cap X_i} x^{(b)}_{i,ij}}{|C_k \cap X_i|} = \frac{n_{kj}^{(i)}}{n_{k+}^{(i)}} = \frac{p_{kj}^{(i)}}{p_{k+}^{(i)}}, \forall k, i, j.
\]

Note that Eq. (2.22) and Eq. (2.23) indicate the specialty of K-means clustering on incomplete data; that is, the centroid of cluster \(C_k\) \((1 \leq k \leq K)\) has no longer existed physically, but rather serves as a “virtual” one just for the computational purpose. It is replaced by the loose combination of \(r\) sub-centroids computed separately on \(r\) IBPs.
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

To minimize $F$, the two-phase iteration process of K-means turns into: (1) Assign $x_i^{(b)}$ ($1 \leq l \leq n$) to the cluster with the smallest distance $f$ computed by Eq. (2.20); (2) Update the centroid of cluster $C_k$ ($1 \leq k \leq K$) by Eq. (2.23). For the convergence of the adjusted K-means, we have the following theorem:

**Theorem 2.3.1** For the objective function given in Eq. (2.21), K-means clustering using $f$ in Eq. (2.20) as the distance function and using $m_{k,i}$, $\forall k, i$, in Eq. (2.23) as the centroid, is guaranteed to converge in finite iterations.

We then extend KCC to the IBP case. Let $P_k^{(i)} = \langle p_k^{(i)} / p_{k+1}^{(i)}, \cdots, p_{kK}^{(i)} / p_{k+1}^{(i)} \rangle = m_{k,i}$, we then have a theorem as follows:

**Theorem 2.3.2** $U$ is a KCC utility function, if and only if $\forall \Pi = \{\pi_1, \cdots, \pi_r\}$ and $K \geq 2$, there exists a set of continuously differentiable convex functions $\{\mu_1, \cdots, \mu_r\}$ such that

$$U(\pi, \pi_i) = p(i) \sum_{k=1}^{K} p_k^{(i)} \mu_i(P_k^{(i)}), \forall i.$$  \hspace{1cm} (2.24)

The convex function $\phi_i$ ($1 \leq i \leq r$), for the corresponding K-means clustering is given by

$$\phi_i(m_{k,i}) = w_i \nu_i(m_{k,i}), \forall k,$$  \hspace{1cm} (2.25)

where

$$\nu_i(x) = a \mu_i(x) + c_i, \forall i, a \in \mathbb{R}_{++}, c_i \in \mathbb{R}.$$  \hspace{1cm} (2.26)

**Remark 5.** The proof is similar to the one for Theorem 2.2.1, so we omit it here. Eq. (2.24) is very similar to Eq. (2.13) except for the appearance of the parameter $p^{(i)}$ ($1 \leq i \leq r$). This parameter implies that the basic partition on a larger data subset will have more impact on the consensus clustering, which is considered reasonable. Also note that when the incomplete data case reduces to the normal case, Eq. (2.24) reduces to Eq. (2.13) naturally. This implies that the incomplete data case is a more general scenario in essence.

### 2.4 Experimental Results

In this section, we present experimental results of K-means-based consensus clustering on various real-world data sets. Specifically, we will first demonstrate the execution efficiency and clustering quality of KCC, and then explore the major factors that affect the performance of KCC. Finally, we will showcase the effectiveness of KCC on handling incomplete basic partitions.
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

Table 2.5: Some Characteristics of Real-World Data Sets

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>Source</th>
<th>#Objects</th>
<th>#Attributes</th>
<th>#Classes</th>
<th>MinClassSize</th>
<th>MaxClassSize</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>breast_w</td>
<td>UCI</td>
<td>699</td>
<td>9</td>
<td>2</td>
<td>241</td>
<td>458</td>
<td>0.439</td>
</tr>
<tr>
<td>ecoli†</td>
<td>UCI</td>
<td>332</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>143</td>
<td>0.899</td>
</tr>
<tr>
<td>iris</td>
<td>UCI</td>
<td>150</td>
<td>4</td>
<td>3</td>
<td>50</td>
<td>50</td>
<td>0.000</td>
</tr>
<tr>
<td>pendigits</td>
<td>UCI</td>
<td>10992</td>
<td>16</td>
<td>10</td>
<td>1055</td>
<td>1144</td>
<td>0.042</td>
</tr>
<tr>
<td>satimage</td>
<td>UCI</td>
<td>4435</td>
<td>36</td>
<td>6</td>
<td>415</td>
<td>1072</td>
<td>0.425</td>
</tr>
<tr>
<td>dermatology</td>
<td>UCI</td>
<td>358</td>
<td>33</td>
<td>6</td>
<td>20</td>
<td>111</td>
<td>0.509</td>
</tr>
<tr>
<td>wine‡</td>
<td>UCI</td>
<td>178</td>
<td>13</td>
<td>3</td>
<td>48</td>
<td>71</td>
<td>0.194</td>
</tr>
<tr>
<td>mm</td>
<td>TREC</td>
<td>2521</td>
<td>126373</td>
<td>2</td>
<td>1133</td>
<td>1388</td>
<td>0.143</td>
</tr>
<tr>
<td>reviews</td>
<td>TREC</td>
<td>4069</td>
<td>126373</td>
<td>5</td>
<td>137</td>
<td>1388</td>
<td>0.640</td>
</tr>
<tr>
<td>la12</td>
<td>TREC</td>
<td>6279</td>
<td>31472</td>
<td>6</td>
<td>521</td>
<td>1848</td>
<td>0.503</td>
</tr>
<tr>
<td>sports</td>
<td>TREC</td>
<td>8580</td>
<td>126373</td>
<td>7</td>
<td>122</td>
<td>3412</td>
<td>1.022</td>
</tr>
</tbody>
</table>

†: two clusters containing only two objects were deleted as noise.
‡: the values of the last attribute were normalized by a scaling factor 100.

2.4.1 Experimental Setup

Experimental data. In the experiments, we used a testbed consisting of a number of real-world data sets obtained from both the UCI and TREC repositories. Table 5.2 shows some important characteristics of these data sets, where “CV” is the Coefficient of Variation statistic [42] that characterizes the degree of class imbalance. A higher CV value indicates a more severe class imbalance.

Validation measure. Since the class labels were provided for each data set, we adopted the normalized Rand index ($R_n$), a long-standing external measure for objective cluster validation. In the literature, it has been recognized that $R_n$ is particularly suitable for K-means clustering evaluation [43]. The value of $R_n$ typically varies in [0,1] (might be negative for extremely poor results), and a larger value indicates a higher clustering quality. More details of $R_n$ can be found in Ref. [14].

Clustering tools. Three types of consensus clustering methods, namely the K-means-based algorithm (KCC), the graph partition algorithm (GP), and the hierarchical algorithm (HCC), were employed in the experiments for the comparison purpose. GP is actually a general concept of three benchmark algorithms: CSPA, HGPA and MCLA [1], which were coded in the MATLAB language and provided by Strehl [3]. HCC is essentially an agglomerative hierarchical clustering algorithm based on the so-called co-association matrix. It was implemented by ourselves in MATLAB following the algorithmic description in Ref. [6]. We also implemented KCC in MATLAB, which includes ten

---

2GP and GCC are interchangeably used.
3Available at: http://www.strehl.com.
utility functions, namely \( U_c, U_H, U_{\text{cos}}, U_{L_5} \) and \( U_{L_8} \), and their corresponding normalized versions (denoted as \( NU_x \)).

To generate basic partitions (BPs), we used the \texttt{kmeans} function of MATLAB with squared Euclidean distance for UCI data sets and with cosine similarity for text data sets. Two strategies, i.e., Random Parameter Selection (RPS) and Random Feature Selection (RFS) proposed in Ref. \([1]\), were used to generate BPs. For RPS, we randomized the number of clusters within an interval for each basic clustering. For RFS, we randomly selected partial features for each basic clustering.

Unless otherwise specified, the default settings for the experiments are as follows. The number of clusters for KCC is set to the number of true clusters (namely the clusters indicated by the known class labels). For each data set, 100 BPs are typically generated for consensus clustering (namely \( r = 100 \)), and the weights of these BPs are exactly the same, i.e., \( w_i = w_j, \forall i, j \). RPS is the default generation strategy for BPs, with the number of clusters for \texttt{kmeans} being randomized within \([K, \sqrt{n}]\), where \( K \) is the number of true clusters and \( n \) is the total number of data objects. When RFS is used instead, we typically select two features randomly for each BP, and set the number of clusters to \( K \) for \texttt{kmeans}. For each \( \Pi \), KCC and GP are run ten times to obtain the average result, whereas HCC is run only once due to its deterministic nature. In each run of KCC, K-means subroutine is called ten times to return the best result. Similarly, in each run of GP, CSPA, HGPA and MCLA are called simultaneously to find the best result.

All experiments in this chapter were run on a Windows 7 platform of SP2 32-bit edition. The PC has an Intel Core i7-2620M 2.7GHz*2 CPU with a 4MB cache, and a 4GB DDR3 664.5MHz RAM.

### 2.4.2 Clustering Efficiency of KCC

The primary concern about a consensus clustering method is usually the efficiency issue. Along this line, We first examine the convergence of KCC, and then its efficiency compared with other methods.
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

Table 2.7: KCC Clustering Results (by $R_n$)

<table>
<thead>
<tr>
<th></th>
<th>$U_c$</th>
<th>$U_H$</th>
<th>$U_{cos}$</th>
<th>$U_{L5}$</th>
<th>$U_{L8}$</th>
<th>$NU_c$</th>
<th>$NU_H$</th>
<th>$NU_{cos}$</th>
<th>$NU_{L5}$</th>
<th>$NU_{L8}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>breast_w</td>
<td>0.0556</td>
<td>0.8673</td>
<td>0.1111</td>
<td>0.1212</td>
<td>0.1333</td>
<td>0.0380</td>
<td>0.8694</td>
<td>0.1173</td>
<td>0.1329</td>
<td>0.1126</td>
</tr>
<tr>
<td>ecoli</td>
<td>0.5065</td>
<td>0.4296</td>
<td>0.4359</td>
<td>0.4393</td>
<td>0.4284</td>
<td>0.5012</td>
<td>0.5470</td>
<td>0.4179</td>
<td>0.4174</td>
<td>0.4281</td>
</tr>
<tr>
<td>iris</td>
<td>0.7352</td>
<td>0.7338</td>
<td>0.7352</td>
<td>0.7352</td>
<td><strong>0.7455</strong></td>
<td>0.7325</td>
<td>0.7069</td>
<td><strong>0.7455</strong></td>
<td><strong>0.7455</strong></td>
<td>0.7352</td>
</tr>
<tr>
<td>pendigits</td>
<td>0.5347</td>
<td>0.5596</td>
<td>0.5692</td>
<td>0.5527</td>
<td>0.5060</td>
<td>0.5652</td>
<td>0.5789</td>
<td>0.5684</td>
<td>0.5639</td>
<td></td>
</tr>
<tr>
<td>satimage</td>
<td>0.4501</td>
<td>0.4743</td>
<td>0.5322</td>
<td>0.4738</td>
<td>0.4834</td>
<td>0.3349</td>
<td><strong>0.5323</strong></td>
<td>0.5318</td>
<td>0.4691</td>
<td>0.4797</td>
</tr>
<tr>
<td>dermatology</td>
<td>0.0352</td>
<td>0.0661</td>
<td>0.0274</td>
<td>0.0223</td>
<td>0.0386</td>
<td>0.0537</td>
<td>0.0490</td>
<td>0.0259</td>
<td>0.0309</td>
<td></td>
</tr>
<tr>
<td>wine</td>
<td>0.1448</td>
<td>0.1476</td>
<td>0.1448</td>
<td>0.1397</td>
<td>0.1379</td>
<td>0.1448</td>
<td>0.1336</td>
<td>0.1449</td>
<td>0.1447</td>
<td>0.1379</td>
</tr>
<tr>
<td>mm</td>
<td>0.5450</td>
<td>0.5702</td>
<td>0.5674</td>
<td>0.5923</td>
<td><strong>0.6184</strong></td>
<td>0.4841</td>
<td>0.5648</td>
<td>0.6023</td>
<td>0.6131</td>
<td><strong>0.6184</strong></td>
</tr>
<tr>
<td>reviews</td>
<td>0.3767</td>
<td>0.4628</td>
<td>0.4588</td>
<td>0.4912</td>
<td>0.4648</td>
<td>0.3257</td>
<td>0.4938</td>
<td>0.5200</td>
<td>0.4817</td>
<td><strong>0.5323</strong></td>
</tr>
<tr>
<td>la12</td>
<td>0.3455</td>
<td><strong>0.3878</strong></td>
<td>0.3647</td>
<td>0.3185</td>
<td>0.3848</td>
<td>0.3340</td>
<td>0.3619</td>
<td>0.3323</td>
<td>0.3451</td>
<td>0.4135</td>
</tr>
<tr>
<td>sports</td>
<td>0.3211</td>
<td>0.4039</td>
<td>0.3429</td>
<td>0.3720</td>
<td>0.3312</td>
<td>0.2787</td>
<td><strong>0.4093</strong></td>
<td>0.3407</td>
<td>0.3053</td>
<td>0.3130</td>
</tr>
</tbody>
</table>

Figure 2.1: Illustration of KCC Convergence with different utility functions.

We generated one set of basic partitions for each data set, and then ran KCC on each $\Pi$ using different utility functions. The average numbers of iterations for convergence are shown in Fig. 2.1. As can be seen, KCC generally converges within 15 iterations regardless of utility functions used, with the only exceptions on pendigits and satimage data sets. Among the ten utility functions, $NU_H$ exhibits the fastest speed of convergence on nearly all data sets except pendigits and satimage, as indicated by the blue solid-line in bold.

We then compare KCC with GP and HCC in terms of execution efficiency. Note that the three methods were run with default settings, and $U_c$ was selected for KCC since it showed a moderate convergence speed in Fig. 2.1 (as indicated by the red dashed-line in bold). Table 2.6 shows the runtime comparison of the three methods, where the fastest one is in bold for each data set.
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

As can be seen, although KCC was run $10 \times 10 = 100$ times for each data set, it is still the fastest one on nine out of eleven data sets. For the large-scale data sets, such as satimage and reviews, the advantage of KCC is particularly evident. HCC seems more suitable for small data sets, such as iris and wine, and becomes struggled for large-scale data sets for its $n$-squared complexity. GP consumes much less time than HCC on large-scale data sets, but it suffers from the high space complexity — that is why it failed to deliver results for three data sets in Table 2.6 (marked as “N/A”), even one more time than HCC. Note that the execution time for generating basic partitions was not included in the table for clarity.

To sum up, KCC shows significantly higher clustering efficiency than the other two popular methods. This is particularly important for real-world applications with large-scale data sets.

2.4.3 Clustering Quality of KCC

Here, we demonstrate the cluster validity of KCC by comparing it with the well-known GP and HCC algorithms. The normalized Rand index $R_n$ was adopted as the cluster evaluation measure.

We took RPS as the strategy for basic partition generation, and employed KCC, GP and HCC with default settings for all the data sets. The clustering results of KCC with different utility functions are shown in Table 2.7. As can be seen, seven out of ten utility functions performed the best on at least one data set, as highlighted in bold. This implies that the diversity of utility functions is very important to the success of consensus clustering. KCC thus achieves an edge by providing a flexible framework that can incorporate various utility functions for different applications. For instance, for the breast_w data set, using $U_H$ or $NU_H$ can obtain an excellent result with $R_n > 0.85$; but the clustering quality will drop sharply to $R_n < 0.15$ if other functions are used instead.

In real-world applications, however, it is hard to know which utility function is the best to a given data set without providing external information. One solution is to rank the utility functions empirically on a testbed, and then set the one with the robust performance as the default choice. To this end, We score a utility function $U_i$ by score$(U_i) = \sum_j \frac{R_n(U_i, D_j)}{\max_i R_n(U_i, D_j)}$, where $R_n(U_i, D_j)$ is the $R_n$ score of the clustering result generated by applying $U_i$ on data set $D_j$. The row “score” at the bottom of Table 2.7 shows the final scores of all the utility functions. As can be seen, the highest score was won by $NU_H$, closely followed by $U_H$, and then $NU_{cos}$ and $U_{cos}$. Since $NU_H$ also showed fast convergence in the previous section, we hereby take it as the default choice for KCC in the experiments to follow. It is also interesting to note that though being the first utility function proposed in the literature, $U_c$ and its normalized version generally perform the worst among all the
listed utility functions.

We then compare the clustering qualities between KCC and the other two methods, where $NU_H$ was selected for KCC. Fig. 2.2 shows the comparative results. Note that in the left (right) sub-graph we omitted three (two) data sets, on which GP (HCC) failed due to the out-of-memory error. As can be seen, compared with GP and HCC, KCC generally shows comparable clustering performance. Indeed, KCC outperformed GP on five out of eight data sets, and beat HCC on five out of nine data sets. Another two observations are also noteworthy. First, although HCC seems closer to KCC in terms of clustering quality, its robustness is subject to doubt — it performed extremely poorly (with a near-to-zero $R_n$ value) on $mm$. Second, the $wine$ and $dermatology$ data sets are really big challenges to consensus clustering — the clustering quality measured by $R_n$ is always below 0.2 no matter what method is used. We will show how to handle this below.

In summary, KCC is competitive to GP and HCC in terms of clustering quality. Among the ten utility functions, $NU_H$ shows more robust performance and thus becomes the primary choice for KCC.

### 2.4.4 Exploration of Impact Factors

In this section, we explore the factors that might affect the clustering performance of KCC. We are concerned with the following characteristics of basic partitions (BPs) in $\Pi$: the number of BPs ($r$), the quality of BPs, the diversity of BPs, and the generation strategy for BPs. Four data sets, i.e., $breast_w$, $iris$, $mm$ and $reviews$, were frequently used as illustrative examples here.
2.4.4.1 Factor I: Number of Basic partitions

To test the impact of the number of BPs, we did random sampling on $\Pi$ (containing 100 BPs for each data set), and generated the subset $\Pi^r$, with $r = 10, 20, \cdots, 90$. For each $r$, we repeated sampling 100 times, and then ran KCC on each sample to get the clustering results, as illustrated by the boxplot in Fig. 2.3.

As can be seen from Fig. 2.3, the volatility of the clustering quality tends to be reduced with the increase of $r$. When $r \geq 50$, the volatility seems to be fixed to a very small interval. This implies that $r = 50$ might be a rough critical point for obtaining robust KCC results in real-world applications. To validate this, we further enlarged the size of the complete set $\Pi$ to 500, and iterated the above experiments by setting $r = 10, 20, \cdots, 490$, respectively. Again we found that the clustering quality of KCC became stable when $r \geq 50$. It is worth noting that this is merely an empirical estimation; the critical point might be raised as the scale of data sets (i.e., $n$) gets significantly larger. Nevertheless, it is no doubt that increasing the number of BPs can effectively suppress the volatility of KCC results.
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

In the field of supervised learning, researchers have long recognized that both the quality and diversity of single classifiers are crucial to the success of an ensemble classifier. Analogously, one may expect that the quality and diversity of basic partitions might affect the performance of consensus clustering. While there have been some initial studies along this line, few research has clearly justified the impact of these two factors based on real-world data experiments. Hardly can we find any research that addressed how they interact with each other in consensus clustering. These indeed motivated our experiments below.

Fig. 2.4 depicts the quality distribution of basic partitions of each data set. As can be seen, the distribution generally has a long right tail, which indicates that there is only a very small portion of BPs that are in relatively high quality. For example, breast_w has four BPs with $R_n$ values over 0.7, but the rests are all below 0.4 and lead to an average: $\overline{R_n} = 0.2240$. Fig. 2.5 then illustrates the pair-wise similarity in $R_n$ between any two BPs. Intuitively, a more diversified $\Pi$ will correspond to a darker similarity matrix, and vice versa. In this sense, the BPs of breast_w and iris are better in diversity than the BPs of mm and reviews. Note that the BPs in each subplot were sorted in the
Based on the above observations, we have the following conjecture: (1) Quality factor: The clustering quality of KCC is largely determined by the small number of BPs in relatively high quality (denoted as HQBP); (2) Diversity factor: The diversity of BPs will become the dominant factor when HQBPs are unavailable. We adopted a stepwise deletion strategy to verify the conjecture. That is, for the set of BPs of each data set, we first sorted BPs in the decreasing order of $R_n$, and then removed BPs gradually from top to bottom to observe the change of clustering quality of KCC. The red solid-lines in Fig. 2.6 exhibit the results. For the purpose of comparison, we also sorted BPs in the increasing order of $R_n$ and repeated the stepwise deletion process. The results are represented by the blue dashed-lines in Fig. 2.6.

As can be seen from the red solid-lines in Fig. 2.6, the KCC quality suffers a sharp drop after removing the first few HQBPs from each data set. In particular, for the three data sets showing more significant long tails in Fig. 2.4, i.e., breast_w, mm and reviews, the quality deterioration is more evident. This implies that it is the small portion of HQBPs rather than the complete set of BPs

---

Figure 2.6: Performance of KCC Based on Stepwise Deletion Strategy.
that determines the quality of KCC. We can verify this point by further watching the variation of the blue dashed-lines, where the removal of BPs in relatively low quality (denoted as LQBP) shows hardly any influence to the KCC quality. Since the removal of LQBP also means the shrinkage of diversity, we can understand that the quality factor represented by the few HQBPs is actually more important than the diversity factor. Therefore, our first conjecture holds. This result also indicates that KCC is capable of taking advantage of a few HQBPs to deliver satisfactory results, even if the whole set of BPs is generally in poor quality.

We then explore the diversity factor by taking a closer look at Fig. 2.6. It is interesting to see that the quality drop of breast \( w \) occurs after removing roughly the first twenty HQBPs, among which only four HQBPs have \( R_n > 0.7 \) (as indicated by Fig. 2.4). This implies that it is the power of the diversity factor that keeps the KCC quality staying in a certain level until too many HQBPs are gone. Furthermore, it is noteworthy from Fig. 2.6 that mm and reviews experience quality drops earlier than breast \( w \) and iris. To understand this, let us recall Fig. 2.5, where the bottom-left areas in much lighter colors indicate that mm and reviews have much poorer diversity than breast \( w \) and iris on LQBP. This further illustrates the existence and the importance of the diversity factor, especially when HQBPs are hard to attain.

In summary, the quality and diversity of basic partitions are both critical to the success of KCC. As the primary factor, the quality level usually depends on a few BPs in relatively high quality. The diversity will become a dominant factor instead when HQBPs are not available.

2.4.4.3 Factor IV: The Generation Strategy of Basic partitions

So far we relied solely on RPS to generate basic partitions, with the number of clusters \( K_i \) \((1 \leq i \leq r)\) varied in \([K, \sqrt{n}]\), where \( K \) is the number of true clusters and \( n \) is the number of data objects. This led to some poor clustering results, such as on UCI data sets wine and dermatology with \( R_n < 0.15 \) and on text data sets la12 and sports with \( R_n \approx 0.4 \) (as shown by Table 2.7 and Fig. 2.2). Here we demonstrate how to use other generation strategies to improve the clustering quality.

We first consider data sets la12 and sports. These are two text data sets with relatively large scales, which means the interval \([K, \sqrt{n}]\) might be too large to generate good-enough BPs. To address this, we still use RPS, but narrow the interval of \( K_i \) to \([2, 2K]\). Fig. 2.7 shows the comparative result. As can be seen, the clustering performance of KCC on the two data sets are improved substantially after the interval adjustment. This clearly demonstrates that KCC might benefit from adjusting RPS when dealing with large-scale data sets. Fig. 2.8 then illustrates the reason for the improvements — a
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

We then illustrate the benefits from using RFS instead of RPS. We employed KCC with RFS on four data sets: *wine*, *dermatology*, *mm*, and *reviews*. For each data set, we gradually increased the number of attributes used to generate basic partitionings (denoted as $d$), and traced the trend of the clustering performance with the increase of $d$. Fig. 2.9 shows the results, where the red dashed-line serves as a benchmark that indicates the original clustering quality using RPS. As can be seen, RFS achieves substantial improvements on *wine* and *dermatology* when $d$ is very small. For instance, the clustering quality on *wine* reaches $R_n = 0.8$ when $d = 2$; it then suffers a sharp fall as $d$ increases, and finally deteriorates to the poor level as RPS, i.e., $R_n < 0.2$, when $d \geq 7$. Similar situation holds for *dermatology*, where KCC with RFS obtains the best clustering results when $5 \leq d \leq 12$. We also tested the performance of RFS on two high-dimensional text data sets *mm* and *reviews*, as shown in Fig. 2.9, where the ratio of the selected attributes increases gradually from 10% to 90%. The results are still very positive — KCC with RFS leads to consistently higher
clustering qualities than KCC with RPS.

Fig. 2.10 takes wine as an example (d = 2) to illustrate the reasons for the improvements. As can be seen, both the quality and diversity of BPs have been improved substantially after employing RFS instead of RPS. Actually, if we further explore wine, we can find that it contains at least five noisy attributes with extremely low $\chi^2$ values. RFS might omit these attributes and thus generate some basic partitions in much higher quality.
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

Given the above experiments, we can understand that the generation strategy of basic partitions has great impact to the clustering performance of KCC. RPS with default settings can serve as the primary choice for KCC to gain better diversity, but should be subject to adjustments when dealing with large-scale data sets. RFS is a good alternative to RPS, especially for data sets on which RPS performs poorly, such as the ones with severe noise in features.

2.4.5 Performances on Incomplete Basic partitions

Here, we demonstrate the effectiveness of KCC on handling basic partitions with missing data. To this end, we adopted two strategies to generate incomplete basic partitions (IBPs). Strategy-I is to randomly remove some data instances from a data set first, and then employ $k$means on the incomplete data set to generate an IBP. Strategy-II is to employ $k$means on the whole data set first, and then randomly remove some labels from the complete basic partition to get an incomplete one. Four data sets, i.e., $\text{breast}_w$, $\text{iris}$, $\text{mm}$ and $\text{reviews}$, were used for KCC with default settings. The removal ratio, denoted as $rr$, was set from 0% to 90% to watch the variation trend.

Figure 2.11: Performances of KCC on Basic partitions with Missing Data.

Given the above experiments, we can understand that the generation strategy of basic partitions has great impact to the clustering performance of KCC. RPS with default settings can serve as the primary choice for KCC to gain better diversity, but should be subject to adjustments when dealing with large-scale data sets. RFS is a good alternative to RPS, especially for data sets on which RPS performs poorly, such as the ones with severe noise in features.

2.4.5 Performances on Incomplete Basic partitions

Here, we demonstrate the effectiveness of KCC on handling basic partitions with missing data. To this end, we adopted two strategies to generate incomplete basic partitions (IBPs). Strategy-I is to randomly remove some data instances from a data set first, and then employ $k$means on the incomplete data set to generate an IBP. Strategy-II is to employ $k$means on the whole data set first, and then randomly remove some labels from the complete basic partition to get an incomplete one. Four data sets, i.e., $\text{breast}_w$, $\text{iris}$, $\text{mm}$ and $\text{reviews}$, were used for KCC with default settings. The removal ratio, denoted as $rr$, was set from 0% to 90% to watch the variation trend.
CHAPTER 2. K-MEANS-BASED CONSENSUS CLUSTERING

Fig. 2.11 shows the result. To our surprise, IBPs generally exert little negative impact to the performance of KCC until \( rr > 70\% \). For the \( mm \) data set, the clustering quality of KCC is even improved from around 0.6 to over 0.8 when \( 10\% \leq rr \leq 70\% \)!

This result strongly indicates that KCC is very robust to the incompleteness of basic partitions; or in other words, it can help to recover the whole cluster structure based on the cluster fragments provided by IBPs. It is also interesting to see that the effect of Strategy-I seems to be comparable to the effect of Strategy-II. In some cases, Strategy-I even leads to better performance than Strategy-II, e.g., on the \( mm \) data set, or on the \( breat \), \( w \) and \( iris \) data sets when \( rr \) is sufficiently large. This observation is somewhat unexpected, since Strategy-II was thought to better reserve the information about the whole cluster structure. This observation, however, is more valuable, since Strategy-I gets more closer to the real-life situations KCC will face in practice.

In summary, KCC shows its robustness in handling incomplete basic partitions. This further validates its effectiveness for real-world applications.

2.5 Concluding Remarks

In this chapter, we established the general theoretical framework of K-means-based consensus clustering (KCC) and provided the corresponding algorithm. We also extended the scope of KCC to the cases where there exist incomplete basic partitions. Experiments on real-world data sets have demonstrated that KCC has high efficiency and the comparable clustering performance with state-of-the-art methods. In particular, KCC has shown robust performances even if only a few high-quality basic partitions are available or the basic partitions have severe incompleteness.
Chapter 3

Spectral Ensemble Clustering

In this chapter, we focus on another category of consensus clustering. The co-association matrix-based methods form a landmark, where a co-association matrix is constructed to summarize basic partitions via measuring how many times a pair of instances occur simultaneously in a same cluster. The main contribution of these methods is the redefinition of the consensus clustering problem as a classical graph partition problem on the co-association matrix, so that agglomerative hierarchical clustering, spectral clustering, or other algorithms can be employed directly to find the consensus partition. It has been well informed that the co-association matrix-based methods can achieve excellent performances \([6, 45]\), but they also suffer from some non-ignorable drawbacks. Particularly, the high time and space complexities prevent it from handling real-life large-scale data, and no explicit global objective function to guide consensus learning might lead to consensus partitions of unstable qualities when facing data sets of different characteristics.

In light of this, we propose Spectral Ensemble Clustering (SEC), which conducts spectral clustering on the co-association matrix to find the consensus partition. Our main contributions are summarized as follows. First, we formally prove that the spectral clustering of a co-association matrix is equivalent to the weighted K-means clustering of a binary matrix, which decreases the time and space complexities of SEC dramatically to roughly linear ones. Second, we derive the intrinsic consensus objective for SEC, which to our best knowledge is the first to give explicit global objective function to a co-association matrix based method, and thus could give clues to its theoretical foundation. Third, we prove theoretically the fine properties of SEC, including its robustness, generalizability and convergence, which are further verified empirically by extensive experiments. Fourth, we extend SEC so as to adapt to incomplete basic partitions, which enables a row-segmentation scheme suitable for big data clustering. Experimental results on various real-world
data sets in both ensemble and multi-view clustering scenarios demonstrate that SEC outperforms some state-of-the-art baselines by delivering higher quality consensus partitions in an efficient way. Besides, SEC is very robust to incomplete basic partitions with many missing values. Finally, the promising ability of SEC in big data clustering is validated via a whole day collection of Weibo data.

### 3.1 Spectral Ensemble Clustering

Let \( X = \{x_1, \ldots, x_n\}^\top \in \mathbb{R}^{n \times d} \) represent the data matrix containing \( n \) instances in \( d \) dimensions. \( \pi_i \) is a crisp basic partition of \( X \) with \( K_i \) clusters generated by some traditional clustering algorithm, and \( \pi_i(x) \in \{1, 2, \cdots, K_i\} \) represents the cluster label of instance \( x \). Given \( r \) basic partitions of \( X \) in \( \Pi = \{\pi_1, \pi_2, \cdots, \pi_r\} \), a co-association matrix \( S_{n \times n} \) is defined as follows [6]:

\[
S(x, y) = \sum_{i=1}^{r} \delta(\pi_i(x), \pi_i(y)), \quad \delta(a, b) = \begin{cases} 
1, & \text{if } a = b \\
0, & \text{if } a \neq b
\end{cases}
\]

In essence, the co-association matrix measures the similarity between each pair of instances, which is the co-occurrence counts of two instances in the same cluster in \( \Pi \).

**Spectral Ensemble Clustering (SEC)** applies spectral clustering on the co-association matrix \( S \) for the final consensus partition \( \pi \), which is formulated as follows:

Let \( H = [H_1, \cdots, H_K] \), a \( n \times K \) partition matrix, be the 1-of-\( K \) coding of \( \pi \), where \( K \) is the user-specified cluster number. The objective function of normalized-cut spectral clustering of \( S \) is the following trace maximization problem:

\[
\max_Z \frac{1}{K} \text{tr}(Z^\top D^{-1/2}S D^{-1/2} Z), \quad \text{s.t. } Z^\top Z = I,
\]

where \( D \) is a diagonal matrix with \( D_{ll} = \sum_q S_{lq}, 1 \leq l, q \leq n \), and \( Z = D^{1/2}H^\top DH^{-1/2} \). A well-known solution to Eq. 3.1 is to run K-means on the top largest \( K \) eigenvectors of \( D^{-1/2}S D^{-1/2} \) for the final consensus partition \( \pi \) [70], which consists of \( K \) cluster \( C_1, C_2, \cdots, C_K \).

#### 3.1.1 From SEC to Weighted K-means

Performing spectral clustering on the co-association matrix, however, suffers from huge time complexity originated from both building the matrix and conducting the clustering. To meet this challenge, one feasible way is to find a more efficient yet equivalent solution for SEC. In what follows, we propose to solve SEC by a weighted K-means clustering on a binary matrix.
CHAPTER 3. SPECTRAL ENSEMBLE CLUSTERING

Let $B_{n \times (\sum_{i=1}^{r} K_i)}$ be a binary matrix derived from the set of $r$ basic partitions in $\Pi$ as follows:

$$
B(x, \cdot) = b(x) = \langle b(x)_1, \cdots, b(x)_r \rangle,
$$

$$
b(x)_i = \langle b(x)_{i1}, \cdots, b(x)_{iK_i} \rangle,
$$

$$
b(x)_{ij} = \begin{cases} 
1, & \text{if } \pi_i(x) = j \\
0, & \text{otherwise}
\end{cases},
$$

where $\langle \cdot \rangle$ indicates a transverse vector. Apparently, $|b(x)_i| = 1$, $\forall i$, where $|\cdot|$ is the $L_1$-norm.

The binary matrix is just to concatenate all the 1-of-$K_i$ codings of basic partitions. Based on $B$, we provide the theorem to connect SEC and classical weighted K-means clustering, from which the calculation of the weights will be also given.

**Theorem 3.1.1** Given $\Pi$, the spectral clustering of $S$ is equivalent to the weighted K-means clustering of $B$; that is,

$$
\max \frac{1}{K} \text{tr}(Z^T D^{-1/2} S D^{-1/2} Z) \Leftrightarrow \sum_{x \in X} f_{m_1, \ldots, m_K}(x),
$$

where $f_{m_1, \ldots, m_K}(x) = \min_k w_b(x) \|b(x)_{wb(x)} - m_k\|^2$, $m_k = \frac{\sum_{x \in C_k} b(x)}{\sum_{x \in C_k} w_b(x)}$, and $w_b(x) = D(x, x) = \sum_{i=1}^{r} \sum_{l=1}^{n} \delta(\pi_i(x), \pi_i(x_l))$.

**Remark 1** By Theorem 3.1.1, we explicitly transform SEC into a weighted K-means clustering in a theoretically equivalent way. Without considering the dimensionality, the time complexity of weighted K-means is roughly $O(I n r K)$, where $I$ is the number of iterations. Thus, the transformation dramatically reduces the time and space complexities from $O(n^3)$ and $O(n^2)$, respectively, to roughly $O(n)$. Note that there is only one non-zero element in $b(x)_i$. Accordingly, while the weighted K-means is conducted on $B$, a $n \times \sum_{i=1}^{r} K_i$ binary matrix, the real dimensionality in computation is merely $r$.

**Remark 2** In Ref. [71], the authors uncovered the connection between spectral clustering and weighted kernel K-means. Differently, for SEC we actually figure out the mapping function of the kernel, which turns out to be the binary data dividing its corresponding weight. By this means, we transform SEC into weighted K-means rather than weighted kernel K-means, which is crucial for gaining high efficiency for SEC and making it practically feasible.
CHAPTER 3. SPECTRAL ENSEMBLE CLUSTERING

Algorithm 1 Spectral Ensemble Clustering (SEC)

Input: \( \Pi = \{ \pi_1, \pi_2, \cdots, \pi_r \} \): \( r \) basic partitions.

\( K \): the number of clusters.

Output: \( \pi \): the consensus partition.

1: Build the binary matrix \( B = [b(x)] \) by Eq. (3.2);
2: Calculate the weight for each instance \( x \) by \( w_b(x) = \sum_{i=1}^{r} \sum_{l=1}^{n} \delta(\pi_i(x), \pi_i(x_l)) \);
3: Call weighted K-means on \( B' = [b(x)/w_b(x)] \) with the weight \( w_b(x) \) and return the partition \( \pi \);

Algorithm 1 gives the pseudocodes of SEC. It is worthy to note that in Line 2, \( \sum_{l=1}^{n} \delta(\pi_i(x), \pi_i(x_l)) \) calculates the size of the cluster where \( x \) belongs to in the \( i \)-th basic partition. Moreover, the binary matrix \( B \) is highly sparse, with only \( r \) non-zero elements existing in each row. The weighted K-means is finally called for the solution.

3.1.2 Intrinsic Consensus Objective Function

By the transformation in Theorem 3.1.1, we give a new insight of the objective function of SEC. Here we derive the intrinsic consensus objective function of SEC to measure the similarity in the partition level. Based on the Table ??, we have the following theorem.

**Theorem 3.1.2** If a utility function takes the form as

\[
U(\pi, \pi_i) = \sum_{k=1}^{K} \frac{n_k}{w_{C_k}} p_k + \sum_{j=1}^{K_i} (\frac{p_{kj}^{(i)}}{p_k})^2, \tag{3.3}
\]

where \( w_{C_k} = \sum_{x \in C_k} w_b(x) \), then it satisfies

\[
\max_{Z} \frac{1}{K} \text{tr}(Z^T D^{-1/2} S D^{-1/2} Z) \Leftrightarrow \max_{\pi} \sum_{i=1}^{r} U(\pi, \pi_i). \tag{3.4}
\]

**Remark 3** The utility function \( U \) of SEC in Eq. (3.3) actually defines a family of utility functions to supervise the consensus learning process. Obviously, \( g(U) \) also holds, if \( g \) is a strictly increasing function. Compared with the categorical utility function, the utility function \( U \) of SEC enforces the weights of the instances in large clusters in a quite natural way. Recall that the co-association matrix measures the similarity at the instance level; by Theorem 3.1.2 we derive the utility function to measure the similarity at the partition level. This indicates that two kinds of similarities at different levels are essentially inter-convertible, which to the best of our knowledge is the first claim in consensus clustering.
Remark 4 Theorem 3.1.2 gives a way of incorporating the weights of basic partitions into the ensemble learning process as follows:

$$\max_{\pi} \sum_{i=1}^{r} \mu_i U(\pi, \pi_i) \Leftrightarrow \sum_{x \in X} f_{m_1, \ldots, m_K}(x),$$

where $\mu$ is the weight vector of basic partitions, $f_{m_1, \ldots, m_K}(x) = \min_{k} w_b(x) \sum_{i=1}^{r} \mu_i \| b(x)_i - m_{k,i} \|^2$, and $m_{k,i} = \sum_{x \in C_k} b(x)_i / \sum_{x \in C_k} w_b(x)$. By this means, we can extend SEC to incorporate the weights of both instances and basic partitions in the ensemble learning process. In what follows, without loss of generality, we set $\mu_i = 1, \forall i$.

3.2 Theoretical Properties

Here, we analyze the learning ability of SEC by exploiting its robustness, generalizability and convergence in theory.

3.2.1 Robustness

Robustness that measures the tolerance of learning algorithms to perturbations (noise) is a fundamental property for learning algorithms. If a new instance is close to a training instance, a good learning algorithm should make their errors similar. This property of algorithms is formulated as robustness by the following definition [72].

**Definition 2 (Robustness)** Let $X$ be the training example space. An algorithm is $(K, \epsilon(\cdot))$ robust, for $K \in \mathbb{N}$ and $\epsilon(\cdot) : X^n \mapsto \mathbb{R}$, if $X$ can be partitioned into $K$ disjoint sets, denoted by $\{C_i\}_{i=1}^{K}$, such that the following holds for all $X \in X^n, \forall x \in X, \forall x' \in X, \forall i = 1, \ldots, K :$ if $x, x' \in C_i$, then $|f_{m_1, \ldots, m_K}(x) - f_{m_1, \ldots, m_K}(x')| \leq \epsilon(X)$.

We then have Theorem 3.2.1 to measure the robustness of SEC as follows:

**Theorem 3.2.1** Let $\mathcal{N}(\gamma, X, \| \cdot \|_2)$ be a covering number of $X$, which is defined to be the minimal integer $m \in \mathbb{N}$ such that there exist $m$ disks with radius $\gamma$ (measured by the metric $\| \cdot \|_2$) covering $X$. For any $x, x' \in X, \| x - x' \|_2 \leq \gamma$, we define $\| b(x)_i - b(x')_i \|_2 \leq \gamma_i$ and $\| w_{b(x)_i} - w_{b(x')_i} \| \leq \gamma_{w,i}, i = 1, \ldots, r$, where $w_{b(x)_i} = \sum_{i=1}^{n} \delta(\pi_i(x), \pi_i(x_1))$. Then, for any centroids $m_1, \ldots, m_K$ learned by SEC, we obtain SEC is $(\mathcal{N}(\gamma, X, \| \cdot \|_2), 2 \sum_{i=1}^{r} \gamma_{w,i} / r + \sqrt{\sum_{i=1}^{r} \gamma_i^2})$-robust.
CHAPTER 3. SPECTRAL ENSEMBLE CLUSTERING

Remark 5 From Theorem 3.2.1 we can see that even if \{γ_i\} and \{γ_w,i\} might be large due to some instances “poorly” clustered by some basic partitions, the high-quality performance of SEC will be preserved, provided that these instances are “well” clustered by other majorities. This means that SEC could benefit from the ensemble of basic partitions.

3.2.2 Generalizability

A small generalization error leads to a small gap between the expected reconstruction error of the learned partition and that of the target one [73]. The generalizability of SEC is highly dependent on the basic partitions. In what follows, we prove that the generalization bound of SEC can converge quickly and SEC can therefore achieve high-quality clustering with a relatively small number of instances.

Theorem 3.2.2 Let \(\pi\) be the partition learned by SEC. For any independently distributed instances \(x_1, \ldots, x_n\) and \(\delta > 0\), with probability at least \(1 - \delta\), the following holds:

\[
E_x f_{m_1,\ldots,m_K}(x) - \frac{1}{n} \sum_{l=1}^{n} f_{m_1,\ldots,m_K}(x_l) \\
\leq \frac{\sqrt{2\pi r K}}{n} \left( \sum_{l=1}^{n} (w_b(x_l))^{-2} \right)^{\frac{1}{2}} + \frac{\sqrt{8\pi r K}}{\sqrt{n \min_{x \in X} w_b(x)}} \\
+ \frac{\sqrt{2\pi r K}}{n \min_{x \in X} (w_b(x))^2} \left( \sum_{l=1}^{n} (w_b(x_l))^2 \right)^{\frac{1}{2}} + \left( \frac{\ln(1/\delta)}{2n} \right)^{\frac{1}{2}}.
\] (3.5)

Remark 6 Theorem 3.2.2 shows that if the third term of the upper bound goes to zero when \(n\) goes to infinity, the empirical reconstruction error of SEC will reach its expected reconstruction error. So, the convergence of

\[
\frac{\sqrt{2\pi r K}}{n} \left( \sum_{l=1}^{n} (w_b(x_l))^{-2} \right)^{\frac{1}{2}} \frac{1}{\min_{x \in X} (w_b(x))^2}
\]

is a sufficient condition for the convergence of SEC. This sufficient condition is easily achieved by the consistency property of the basic partitions.

Remark 7 The consistency of crisp basic partitions will make \(w_b(x_l)/|C_k|\) diverge little, where \(|C_k|\) denotes the cardinality of the cluster containing \(x_l\). If we further assume that \(|C_k| = a_k n\), where \(a_k \in (0, 1)\), the convergence of SEC can be as fast as \(O(1/\sqrt{n})\). The fast convergence rate will result in the expected risk of the learned partition decreasing quickly to the expected risk of the target partition [74]. This verifies the efficiency of SEC. Compared with classical K-means clustering, the fastest known convergence rate is \(O(1/\sqrt{n})\) [74] [75].
CHAPTER 3. SPECTRAL ENSEMBLE CLUSTERING

3.2.3 Convergence

Due to the good convergence of weighted K-means, SEC will converge w.r.t. $n$. Here, we show that it will also converge w.r.t. $r$, the number of basic partitions, which means that the final clustering $\pi$ will become more robust and stable as we keep increasing the number of basic partitions.

Theorem 3.2.3 \( \forall \lambda > 0, \) there exists a clustering $\pi_0$ such that

$$
\lim_{r \to \infty} \Pr\{|\pi - \pi_0| \geq \lambda\} \to 0,
$$

where $\pi$ is the final consensus clustering output by SEC and $\Pr\{A\}$ denotes the probability of event $A$.

Remark 8 Theorem 3.2.3 implies that the centroids $m_1, \ldots, m_K$ will converge to $m^0_1, \ldots, m^0_K$ as $r$ goes to infinity. Thus, the output of SEC will converge to the true clustering as we increase the number of basic partitions sufficiently.

3.3 Incomplete Evidence

In practice, incomplete basic partitions (IBP) are easily met for data collecting device failures or transmission loss. By clustering a data subset $X_i \subseteq X$, $1 \leq i \leq r$, we can obtain an incomplete basic partition $\pi_i$ of $X$. Assume $r$ data subsets can cover the whole data set, i.e., \( \bigcup_{i=1}^r X_i = X \) with $|X_i| = n^{(i)}$. The problem is how to cluster $X$ into $K$ crisp clusters using SEC given $r$ IBPs in $\Pi = \{\pi_1, \ldots, \pi_r\}$.

Due to the missing values in $\Pi$, the co-association matrix cannot reflect the similarity of instance pairs any longer. To address this challenge, we start from the objective function of weighted K-means and extend it to handling incomplete basic partitions. It is obvious that missing elements in basic partitions provide no utility in the ensemble process. Consequently, they should not be involved in the weighted K-means for the centroid computation. We therefore have:

Theorem 3.3.1 Given $r$ incomplete basic partitions, we have

$$
\sum_{x \in X} f_{m_1, \ldots, m_K} (x) \Leftrightarrow \max \sum_{i=1}^r p^{(i)} \sum_{k=1}^K n_{k+}^{(i)} p_{k+} \sum_{j=1}^K \left( \frac{p_{kj}}{p_{k+}} \right)^2,
$$

where $f_{m_1, \ldots, m_K} (x) = \min_k \sum_{i \in X_i} w_{b(x)} \| b(x)_{i+} - m_{k,i+} \|^2$, with $p^{(i)} = n^{(i)}/n$, $n_{k+}^{(i)} = |C_k \cap X_i|$, $w_{C_k}^{(i)} = \sum_{x \in C_k \cap X_i} w_{b(x)}$, $m_{k,i} = \sum_{x \in C_k \cap X_i} b(x)i/ \sum_{x \in C_k \cap X_i} w_{b(x)}$. 


Remark 9 Compared with Theorem 3.1.2, the utility function of SEC with IBPs has one more parameter \( p^{(i)} \). This indicates that basic partitions with more elements are naturally assigned with higher importance for the ensemble process, which agrees with our intuition. This theorem also demonstrates the advantages of the transformation from co-association matrix to binary matrix; that is, the former cannot reflect the incompleteness of basic partitions while the latter can.

For the convergence of the SEC with IBPs, we have:

**Theorem 3.3.2** For the objective function in Eq. (3.6), SEC with IBPs is guaranteed to converge in finite two-phase iterations of weighted K-means clustering.

**Theorem 3.3.3** SEC with IBPs holds the convergence property as the number of IBPs \( r \) increases.

### 3.4 Towards Big Data Clustering

When it comes to big data, it is often difficult to conduct traditional cluster analysis due to the huge data volume and/or high data dimensionality. Ensemble clustering like SEC with the ability in handling incomplete basic partitions becomes a good candidate towards big data clustering.
CHAPTER 3. SPECTRAL ENSEMBLE CLUSTERING

In order to conduct large-scale data clustering, we propose the so-called row-segmentation strategy. Specifically, to generate each basic partition, we randomly select a data subset with a certain sampling ratio from the whole data, and run K-means on it to obtain an incomplete basic partition; this process repeats \( r \) times, prior to running SEC to obtain the final consensus partition.

The benefit of employing the row-segmentation strategy is two-fold. On one hand, a big data set can be decomposed into several smaller ones, which can be handled independently and separately to obtain IBPs. On the other hand, in the final consensus clustering, no matter how large the dimensionality of the original data is, we only need to conduct weighted K-means on the binary matrix \( B \) with only \( r \) non-zero elements in each row during the ensemble learning process. Note that Ref. [45] made the co-association matrix sparse for a fast decomposition, but we here transform the co-association matrix into the binary matrix directly so that we even do not need to build the co-association matrix. The experimental results in the next section demonstrate that the row-segmentation strategy does work well and even outperforms the basic clustering on the whole data.

3.5 Experimental Results

In this section, we evaluate SEC on abundant real-world data sets of different domains, and compare it with several state-of-the-art algorithms across both ensemble clustering and multi-view clustering areas. In the first scenario, each data set is provided with a single view and basic partitions are produced by some random sampling schemes. In the second scenario, however, each data set is provided with multiple views and each view generates either one or multiple basic partitions by random sampling. Finally, a case study on large-scale Weibo data shows the ability of SEC for big data clustering.

3.5.1 Scenario I: Ensemble Clustering

3.5.1.1 Experimental Setup

Data. Various real-world data sets with true cluster labels are used for evaluating the experiments in the scenario of ensemble clustering. Table 3.1 summarizes some important characteristics of these data sets obtained from UCI\(^1\), CLUTO\(^2\) and LIBSVM\(^3\) repositories, respectively.

\(^1\)https://archive.ics.uci.edu/ml/datasets.html.
\(^2\)http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download.
\(^3\)http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.
CHAPTER 3. SPECTRAL ENSEMBLE CLUSTERING

Tool. SEC is coded in MATLAB. The \texttt{kmeans} function in MATLAB with either squared Euclidean distance (for UCI and LIBSVM data sets) or cosine similarity (for CLUTO data sets) is run 100 times to obtain basic partitions by varying the cluster number in \([K, \sqrt{n}]\), where \(K\) is the true cluster number and \(n\) is the data size. For two relatively large data sets \texttt{letter} and \texttt{mnist}, the cluster numbers of basic partitions vary in \([2, 2K]\) for meaningful partitions. The baseline methods include consensus clustering with category utility function (CCC, a special case of KCC \cite{46}), graph-based consensus clustering methods (GCC, including CSPA, HGPA and MCLA) \cite{1}, co-association matrix with agglomerative hierarchical clustering (HCC with group-average, single-linkage and complete-linkage) \cite{6}, and probability trajectory based graph partitioning (PTGP) \cite{52}. These baselines are selected for the following reasons: GCC has great impacts in the area of consensus clustering; CCC shares common grounds with SEC by employing a K-means-like algorithm; both HCC and PTGP are co-association matrix based methods, and the former is a very famous one and the latter is newly proposed. All the methods are coded in MATLAB and set with default settings. The cluster number for SEC and all baselines is set to the true one for fair comparison. All basic partitions are equally weighted (\(i.e., \mu=1\)). Each algorithm runs 50 times for average results and deviations.

Validation. We employ external measures to assess cluster validity. It is reported that the normalized Rand index (\(R_n\) for short) is theoretically sound and shows excellent properties in practice \cite{43}.

Environment. All experiments in Scenarios I&II were run on a PC with an Intel Core i7-3770 3.4GHz*2 CPU and a 32GB DDR3 RAM.

3.5.1.2 Validation of Effectiveness

Here, we compare the performance of SEC with that of baseline methods in consensus clustering. Table 3.2 (Left side) shows the clustering results, with the best results highlighted in \textit{bold red} and the second best in \textit{italic blue}.

Firstly, it is obvious that SEC shows clear advantages over other consensus clustering baselines, with 10 best and 9 second best results out of the total 19 data sets; in particular, the margins for the three data sets: \texttt{wine}, \texttt{la12} and \texttt{mm} are very impressive. To fully compare the performance of different algorithms, we propose a measurement score as follows: \(\text{score}(A_i) = \sum_j \frac{R_n(A_i, D_j)}{\max_i R_n(A_i, D_j)}\), where \(R_n(A_i, D_j)\) denotes the \(R_n\) value of the \(A_i\) algorithm on the \(D_j\) data set. This score evaluates certain algorithm by the best performance achieved by the state-of-the-art methods. From this score, we can see that SEC exceeds other consensus clustering methods by a large margin.
CHAPTER 3. SPECTRAL ENSEMBLE CLUSTERING

Let us take a close look at HCC, which as SEC also leverages co-association matrix for consensus clustering. It is obvious that SEC outperforms HCC with group-average (HCC\_GA) completely, in 13 out of 19 data sets, although HCC\_GA is already the second best among the baselines. The implication is two-fold: First, the superior performances of SEC and HCC\_GA indicate that the co-association matrix indeed does well in integrating information for consensus clustering; Second, a spectral clustering is much better than a hierarchical clustering in making the most of a co-association matrix. The reason for the second point is complicated, but the lack of explicit global objective function in HCC variants might be one of them; that is, unlike CCC or SEC, HCC variants have no utility function to supervise the process of consensus learning, and therefore could perform much less stably than SEC. This is supported by the extremely poor performances of HCC\_GA on cacmcisi and mm in Table 3.2 with negative $R_n$ values even poorer than that of random labeling. Similar observations can be found for the newly proposed algorithm PTGP on mm, which employs the mini-cluster based core co-association matrix but also lacks of utility functions for consensus learning.
Table 3.2: Clustering Results (by $R_n$) and Running Time (by sec.) in Scenario I

<table>
<thead>
<tr>
<th>Data set</th>
<th>Clustering Result</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SEC</td>
<td>CCC</td>
</tr>
<tr>
<td>breast_w</td>
<td>0.82±0.00</td>
<td>0.07±0.05</td>
</tr>
<tr>
<td>iris</td>
<td>0.92±0.02</td>
<td>0.74±0.02</td>
</tr>
<tr>
<td>wine</td>
<td>0.33±0.00</td>
<td>0.14±0.00</td>
</tr>
<tr>
<td>cancer</td>
<td>0.64±0.02</td>
<td>-0.04±0.00</td>
</tr>
<tr>
<td>classic</td>
<td>0.68±0.02</td>
<td>0.37±0.07</td>
</tr>
<tr>
<td>crammed</td>
<td>0.95±0.00</td>
<td>0.96±0.00</td>
</tr>
<tr>
<td>hitech</td>
<td>0.29±0.01</td>
<td>0.21±0.02</td>
</tr>
<tr>
<td>kib</td>
<td>0.57±0.08</td>
<td>0.32±0.08</td>
</tr>
<tr>
<td>la12</td>
<td>0.51±0.07</td>
<td>0.32±0.10</td>
</tr>
<tr>
<td>mm</td>
<td>0.62±0.05</td>
<td>0.43±0.07</td>
</tr>
<tr>
<td>rel</td>
<td>0.28±0.02</td>
<td>0.27±0.02</td>
</tr>
<tr>
<td>reviews</td>
<td>0.53±0.05</td>
<td>0.43±0.08</td>
</tr>
<tr>
<td>sports</td>
<td>0.47±0.03</td>
<td>0.29±0.08</td>
</tr>
<tr>
<td>tr11</td>
<td>0.59±0.06</td>
<td>0.46±0.07</td>
</tr>
<tr>
<td>tr12</td>
<td>0.46±0.03</td>
<td>0.43±0.04</td>
</tr>
<tr>
<td>tr41</td>
<td>0.45±0.05</td>
<td>0.38±0.05</td>
</tr>
<tr>
<td>tr45</td>
<td>0.45±0.05</td>
<td>0.33±0.04</td>
</tr>
<tr>
<td>letter</td>
<td>0.12±0.01</td>
<td>0.12±0.00</td>
</tr>
<tr>
<td>mnist</td>
<td>0.42±0.02</td>
<td>0.40±0.02</td>
</tr>
<tr>
<td>score/avg.</td>
<td>18.60</td>
<td>12.65</td>
</tr>
</tbody>
</table>

Note: (1) N/A means the out-of-memory failures. (2) We omit the zero standard deviations of CSPA, MCLA, HCC and PTGP for space concern. (3) In runtime comparison, we omit two variants of HCC with similar performances due to space concern. (4) The best is highlighted in bold, and the second best in italic.
We finally turn to CCC, which shares with SEC the K-means clustering in consensus clustering but assigns equal weights to instances. From Table 3.2, the performance of CCC seems much poorer than that of SEC, especially on breast, w and cacmcisi. This indicates that equally weighting of data instances might not be appropriate for consensus learning. In contrast, starting from the spectral clustering view of a co-association matrix, SEC enforces the weights of the instances in large clusters in a quite natural way, and finally leads to superior performances.

3.5.1.3 Validation of Efficiency

Table 3.2 (Right side) shows the average execution time of various consensus clustering methods with 50 repetitions. Since HCC variants have similar execution time, we here only report the results of HCC, GA due to limited space. It is obvious that the K-means-like methods, such as SEC and CCC, get clear edges to competitors, and HCC runs the slowest for adopting hierarchical clustering. This indeed demonstrates the value of SEC in transforming spectral clustering of co-association matrix into weighted K-means clustering. On one hand, we make use of co-association matrix to integrate the information of basic partitions nicely. On the other hand, we avoid generating and handling co-association matrix directly but make use of weighted K-means clustering on the binary matrix to gain high efficiency. Although PTGP runs faster than HCC, it needs much more memory and fails to deliver results for two large data sets letter and mnist.

3.5.1.4 Validation of Robustness

Fig. 3.1(a) and Fig. 3.1(c) demonstrate the robustness of SEC by taking breast, w and cranmed as example. We choose these two data sets due to their relatively well-structured clusters — it is often difficult to observe the theoretical properties of an algorithm given very poor performances.
CHAPTER 3. SPECTRAL ENSEMBLE CLUSTERING

Table 3.3: Experimental Data Sets for Scenario II

<table>
<thead>
<tr>
<th>View</th>
<th>Digit</th>
<th>3-Sources</th>
<th>Multilingual</th>
<th>4-Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pixel (240)</td>
<td>BBC (3560)</td>
<td>English (9749)</td>
<td>Conference (20)</td>
</tr>
<tr>
<td>2</td>
<td>Fourier (74)</td>
<td>Guardian (3631)</td>
<td>German (9109)</td>
<td>Term (13214)</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>Reuters (3068)</td>
<td>French (7774)</td>
<td>-</td>
</tr>
<tr>
<td>#Instances</td>
<td>2000</td>
<td>169</td>
<td>600</td>
<td>4236</td>
</tr>
<tr>
<td>#Classes</td>
<td>10</td>
<td>6</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

We can see that for each data set, the majority of basic partitions are of very low quality. For example, the quality of over 60 basic partitions on cranmed is below 0.1 in terms of $R_n$. Nevertheless, SEC performs excellently (with $R_n > 0.95$) by leveraging the diversity among poor basic partitions. Similar phenomena also occur on some other data sets like breast_w, which indicates the power of SEC in fusing diverse information from even poor basic partitions.

3.5.1.5 Validation of Generalizability and Convergence

Next, we check the generalizability and convergence of SEC. Fig. 3.1(b) and Fig. 3.1(d) show the results by varying the number of basic partitions from 20 to 80 for breast_w and cranmed, respectively. Note that the above process is repeated 20 times for average results. Generally speaking, it is clear that with the increasing number of basic partitions (i.e., $r$), the performance of SEC goes up and becomes stable gradually. For instance, SEC achieves satisfactory result from breast_w with only 20 basic partitions, but it also suffers from high volatility given such a small $r$; when $r$ goes up, the variance becomes narrow and stabilizes in a small region.

3.5.1.6 Effectiveness of Incompleteness Treatment

Here, we demonstrate effectiveness of SEC in handling incomplete basic partitions (IBP). The row-segmentation strategy is employed to generate IBPs. In detail, data instances are firstly randomly sampled with replacement, with the sampling ratio going up from 20% to 80%, to form overlapped data subsets and generate IBPs; SEC is then called to ensemble these IBPs and obtain a consensus partition. Note that for each ratio, the above process repeats 100 times to obtain IBPs, and unsampled instances are omitted in the final consensus learning. It is intuitive that a lower sampling ratio leads to smaller overlaps between IBPs and thus worse clustering performances. Fig. 3.2 shows the sample results on mm and reviews, where the horizontal line indicates the K-means clustering result on the original data set and serves as the baseline unchanged with the sampling ratio. As can
be seen, SEC keeps providing stable and competitive results as the sampling ratio goes down to 20%, which demonstrates the effectiveness of incompleteness treatment of SEC.

3.5.2 Scenario II: Multi-view Clustering

3.5.2.1 Experimental Setup

Data. Four real-world data sets, i.e., UCI Handwritten Digit, 3-Sources, Multilingual and 4-Areas listed in Table 3.3, are used in the experiments. UCI Handwritten Digit consists of 0-9 handwritten digits obtained from the UCI repository, where each digit has 200 instances with 240 features in pixel view and 76 features in Fourier view. 3-Sources is collected from three online news sources: BBC, Guardian and Reuter, from February to April in 2009. Of these documents, 169 are reported in all three sources (views). Each document is annotated with one of six categories: business, entertainment, health, politics, sports and technology. Multilingual contains the documents written originally in five different languages over 6 categories. We here use the sample suggested by [64], which has 100 documents for each category with three views in English, German and French, respectively. 4-Areas is derived from 20 conferences in four areas including database, data mining, machine learning and information retrieval. It contains 28,702 authors and 13,214 terms in the abstract. Each author is labeled with one or multiple areas, and the cross-area authors are removed for unambiguous evaluation. The remainder has 4,236 authors in both conference and term views.

Tool. We compare SEC with a number of baseline algorithms including ConKM, ConNMF, ColNMF [61], CRSC [63], MultiNMF [64] and PVC [65]. All the competitors are with default settings whenever possible. Gaussian kernel is used to build the affinity matrix for CRSC. The trade-off parameter $\lambda$ is set to 0.01 for MultiNMF as suggested in Ref. [64]. For SEC, we employ the kmeans function in MATLAB to generate one basic partition for each view, and then call SEC to fuse them with equal weights into a consensus one. Each algorithm is called 50 times for the average results.

Validation. For consistency, we also employ $R_n$ to evaluate cluster validity.
CHAPTER 3. SPECTRAL ENSEMBLE CLUSTERING

Table 3.4: Clustering Results in Scenario II (by $R_n$)

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Digit</th>
<th>3-Sources</th>
<th>Multilingual</th>
<th>4-Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConKM</td>
<td>0.58±0.06</td>
<td>0.16±0.08</td>
<td>0.12±0.04</td>
<td>0.00±0.00</td>
</tr>
<tr>
<td>ConNMF</td>
<td>0.49±0.06</td>
<td>0.28±0.09</td>
<td>0.22±0.02</td>
<td>0.03±0.06</td>
</tr>
<tr>
<td>ColNMF</td>
<td>0.39±0.03</td>
<td>0.20±0.05</td>
<td>0.22±0.02</td>
<td>0.11±0.14</td>
</tr>
<tr>
<td>CRSC</td>
<td>0.64±0.03</td>
<td>0.30±0.04</td>
<td>0.24±0.01</td>
<td>0.00±0.00</td>
</tr>
<tr>
<td>MultiNMF</td>
<td><strong>0.65±0.03</strong></td>
<td>0.22±0.06</td>
<td>0.22±0.02</td>
<td>0.00±0.00</td>
</tr>
<tr>
<td>PCV</td>
<td>0.56±0.00</td>
<td>N/A</td>
<td>N/A</td>
<td>0.01±0.00</td>
</tr>
<tr>
<td>SEC</td>
<td>0.44±0.05</td>
<td><strong>0.55±0.09</strong></td>
<td><strong>0.25±0.03</strong></td>
<td><strong>0.56±0.09</strong></td>
</tr>
</tbody>
</table>

Note: N/A means no result due to more than two views data sets.

Table 3.5: Clustering Results in Scenario II with pseudo views (by $R_n$)

<table>
<thead>
<tr>
<th>Data sets</th>
<th>Digit</th>
<th>3-Sources</th>
<th>Multilingual</th>
<th>4-Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConKM</td>
<td>0.62±0.09</td>
<td>0.09±0.05</td>
<td>0.15±0.04</td>
<td>0.00±0.00</td>
</tr>
<tr>
<td>ConNMF</td>
<td>0.51±0.05</td>
<td>0.25±0.04</td>
<td>0.21±0.00</td>
<td>0.02±0.06</td>
</tr>
<tr>
<td>ColNMF</td>
<td>0.43±0.07</td>
<td>0.14±0.09</td>
<td>0.20±0.00</td>
<td>0.04±0.08</td>
</tr>
<tr>
<td>CRSC</td>
<td>0.66±0.02</td>
<td>0.32±0.02</td>
<td>0.25±0.04</td>
<td>0.00±0.00</td>
</tr>
<tr>
<td>MultiNMF</td>
<td>0.65±0.06</td>
<td>0.23±0.08</td>
<td>0.22±0.01</td>
<td>0.00±0.01</td>
</tr>
<tr>
<td>PCV</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>SEC</td>
<td><strong>0.69±0.06</strong></td>
<td><strong>0.62±0.09</strong></td>
<td><strong>0.29±0.03</strong></td>
<td><strong>0.67±0.09</strong></td>
</tr>
</tbody>
</table>

Note: N/A means no result due to more than two views data sets.

3.5.2.2 Comparison of Clustering Quality

Table 3.4 shows the clustering results on four multi-view data sets, with the best results highlighted in bold red and the second best in italic blue. The sign “N/A” indicates PVC cannot handle data with more than two views.

As can be seen from Table 3.4, SEC generally shows higher clustering performances than the baselines, especially for data sets 3-Sources and 4-Areas — actually all baselines seem completely ineffective in inferring the structure of 4-Areas. This indeed reveals the unique merit of SEC for multi-view clustering; that is, SEC works on new features from basic partitions rather than original features, which might avoid the negative impact of data dimensionality, especially when dealing with data sets such as 4-Areas that have two views of substantially different dimensionalities.

It is also noteworthy that SEC has poor performance on Digit. If we take a close look at the two basic partitions for SEC, we can find the contrastive performances, i.e., $R_n = 0.65$ and 0.32 on “Pixel” and “Fourier”, respectively. As a result, given the only two basic partitions, SEC can only find a compromise and thus results in poor performance. One straightforward remedy is to make full use of the robustness of SEC by increasing the number of basic partitions in each view, as suggested by Section 3.2.1 and Section 3.5.1.4. We give experimental results below.

46
3.5.2.3 Robustness Revisited

As mentioned above, sufficient basic partitions could enhance the robustness of SEC via repeating valid local structures. To better understand this, in this experiment, we generate \( r = 20 \) basic partitions for each view using a random feature selection scheme. That is, for each basic partition, we take all the data instances but sample the features randomly with a ratio \( r_s \) so as to form a data subset. We set \( r_s = 50% \) empirically for keeping enough feature information for basic clustering yet without sacrificing the diversity of basic partitions. By this means, SEC gains multiple pseudo views of data, which is good to leverage its robustness property.

From Table 3.5, extra pseudo views indeed boost the performance of multi-view clustering and SEC. Specially, the competitive multi-view clustering methods have slightly improvements on the first three data sets while SEC consistently have significant gains on all four data sets. In particular, SEC with pseudo views performs even better than the baselines on Digit. This not only demonstrates the effectiveness of random feature selection for basic partition but also illustrates how to inspire the robustness of SEC in multi-view learning.

3.5.2.4 Dealing with Partial Multi-view Clustering

In real-world applications, it is common to collect partial multi-view data, i.e., incomplete data in different views, due to device failures or transmission loss [65]. Here we validate the performance of SEC on partial multi-view data, with the well-known PVC designed purposefully for partial multi-view clustering as a baseline.

To simulate the partial multi-view setting, we randomly select a fraction of instances, from
CHAPTER 3. SPECTRAL ENSEMBLE CLUSTERING

Table 3.6: Sample Weibo Clusters Characterized by Keywords

<table>
<thead>
<tr>
<th>ID</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clu.3</td>
<td>term begins, campus, partner, teacher, school, dormitory</td>
</tr>
<tr>
<td>Clu.21</td>
<td>Mid-Autumn Festival, September, family, happy, parents</td>
</tr>
<tr>
<td>Clu.40</td>
<td>China, powerful, history, victory, Japan, shock, harm</td>
</tr>
<tr>
<td>Clu.65</td>
<td>Meng Ge, mother, apologize, son, harm, regret, anger</td>
</tr>
<tr>
<td>Clu.83</td>
<td>travel, happy, dream, life, share, picture, plan, haha</td>
</tr>
</tbody>
</table>

5% to 30% with 5% as interval from each view. In Fig. 3.3, the four solid lines in blue represent the performances of SEC on four data sets with varying missing rates, and the two dash lines depict the results of PVC on Digit and 4-Areas. Note that: 1) SEC employs pseudo views with \( r = 20 \) and \( r_s = 50\% \) for each view; 2) PVC only has results on two-view data sets.

From Fig. 3.3 we can see that the performance of SEC and PVC generally goes down as the missing rate increases. Nevertheless, SEC behaviors relatively stably on three-view rather than two-view data sets. This is because three-view data sets can provide more information given the same missing rate on each view. More importantly, SEC outperforms PVC by clear margins in nearly all scenarios on Digit and 4-Areas, which again demonstrates the advantage of SEC in handling incomplete basic partitions.

3.5.3 SEC for Weibo Data Clustering

Sina Weibo\(^8\), a Twitter-like service launched in 2009, is a popular social media platform in China. It has accumulated more than 500 million users and has around 100 million tweets published everyday, which provides tremendous value for commercial applications and academic research.

Next we illustrate how to employ SEC to cluster the entire Weibo data published on Sept. 1st, 2013, which consist of 97,231,274 Chinese tweets altogether. Python environment is adopted to facilitate text processing. After removing 30 million advertisement related tweets via simple keywords filtering, SCWS\(^9\) is applied to build the vector space model with top 10,000 frequently used terms. By this means, we obtain a text corpus with 61,212,950 instances and 10,000 terms. Next, the row-segmentation strategy proposed in Section 3.4 is called to acquire 100 data subsets each with 10,000,000 instances, and the famous text clustering tool CLUTO\(^10\) with default settings is then called in parallel to cluster these data subsets into basic partitions.

---

\(^8\)http://www.weibo.com/.
\(^9\)http://www.xunsearch.com/scws/.
\(^10\)http://glaros.dtc.umn.edu/gkhome/views/cluto.
SEC is finally called to fuse the basic partitions into a consensus one. To achieve this, we build a simple distributed system with 10 servers to accelerate the fusing process. In detail, the binary matrix derived from the 100 IBPs is firstly horizontally split and distributed to every computational nodes. One server is chosen as the master to broadcast the centroid matrix to all nodes during weighted K-means clustering. Each node then computes the distances between local binary vectors and the centroids, assigns the cluster labels, and summarizes a partial centroid matrix as return to the master server. After receiving all partial centroid matrices in the master node, the centroid matrix is updated and a new iteration begins. Note that the cluster number is set to 100 for both basic and consensus clustering.

The results of some clusters tagged by the representative keywords are shown in Table 4. It can be inferred easily that Cluster #3, #21, and #83 represent “the beginning of new semester”, “mid-autumn festival”, and “travel” events, respectively. In Cluster #40, the tweets reflect the user opinions towards the conflict between China and Japan due to the “September 18th incident”; Cluster #65 reports a hot event that Meng Ge, a famous female singer in China, apologized for her son’s crime. In general, although the basic partitions are highly incomplete, some interesting events can still be discovered by using the row-segmentation strategy. SEC appears to be a promising candidate for big data clustering.

3.6 Summary

In this chapter, we proposed the Spectral Ensemble Clustering (SEC) algorithm. By identifying the equivalent relationship between SEC and weighted K-means, we decreased the time and space complexities of SEC dramatically. The intrinsic consensus objective function of SEC was also revealed, which bridges the co-association matrix based methods with the methods with explicit global objective functions. We then investigated the robustness, generalizability and convergence properties of SEC to showcase its superiority in theory, and extended it to handle incomplete basic partitions. Extensive experiments demonstrated that SEC is an effective and efficient algorithm compared with some state-of-the-art methods in both the ensemble and multi-view clustering scenarios. We further proposed a row-segmentation scheme for SEC, and demonstrated its effectiveness via the case of consensus clustering of big Weibo data.
Chapter 4

Infinite Ensemble Clustering

Recently, representation learning attracts substantial research attention, which has been widely adopted as the unsupervised feature pre-treatment \cite{76}. The layer-wise training and the followed deep structure are able to capture the visual descriptors from coarse to fine \cite{77,78}. Notably, there are a few deep clustering methods proposed recently, working well with either feature vectors \cite{79} or graph Laplacian \cite{80,65}, towards high-performance generic clustering tasks. There are two typical problems with regard to the existing deep clustering approaches: (1) how to seamlessly integrate the “deep” concept into the conventional clustering framework, (2) how to solve it efficiently. Few attempts have been made for the first problem \cite{80,81}, however, most of which sacrifice the time efficiency. They follow the conventional training strategy for deep models, whose complexity will be in super-linear with respect to the number of samples. A recent deep linear coding framework attempts to handle the second problem \cite{79}, and preliminary results demonstrate its time efficiency with comparable performance on large-scale data sets. However, its performance on vision data has not been thoroughly evaluated yet, given different visual descriptors and tasks.

Tremendous efforts have been made in ensemble clustering and deep representation, which lead us to wonder whether these two powerful tools can be strongly coupled for the unsolved challenging problems. For example, it has been widely recognized that with the increasing number of basic partitions, ensemble clustering achieves better performance and lower variance \cite{49,82}. However, the best number of basic partitions for a given data sets still remains an open problem. Too few basic partitions cannot exert the capacity of ensemble clustering, while too many basic partitions lead to unnecessary computational resource waste. Here comes the third problem that (3) can we use the infinite ensemble basic partitions to maximize the capacity of ensemble clustering with a low computational cost?
CHAPTER 4. INFINITE ENSEMBLE CLUSTERING

In this work, we simultaneously manage to tackle the three problems mentioned above, and conduct extensive experiments on numerous data sets with different visual descriptors for demonstration. Our new model links the marginalized denoising auto-encoder to ensemble clustering and leads to a natural integration named “Infinite Ensemble Clustering” (IEC), which is simple yet effective and efficient. To that end, we first generate a moderate number of basic partitions, as the basis for the ensemble clustering. Second, we convert the preliminary clustering results from the basic partitions to 1-of-$K$ codings, which disentangles dependent factors among data samples. Then the codings are expanded infinitely by considering the empirical expectation over the noisy codings through the marginalized auto-encoders with the drop-out noises. Two different deep representations of IEC are provided with the linear or non-linear model. Finally, we run K-means on the learned representations to obtain the final clustering. The framework of IEC is demonstrated in Figure 4.1. The whole process is similar to marginalized Denoising Auto-Encoder (mDAE). Several basic partitions are fed into the deep structure with drop-out noises in order to obtain the expectation of the co-association matrix. Extensive results on diverse vision data sets show that our IEC framework works fairly well with different visual descriptors, in terms of time efficiency and clustering performance, and moreover some key impact factors are thoroughly studied as well. The pan-omics gene expression analysis application shows that IEC is a promising tool for real-world multi-view and incomplete data clustering.

We highlight our contributions as follows.

- We propose a framework called Infinite Ensemble Clustering (IEC) which integrates the deep structure and ensemble clustering. By this means, the complex ensemble clustering problem
CHAPTER 4. INFINITE ENSEMBLE CLUSTERING

can be solved with a stacked marginalized Denoising Auto-Encoder structure in an efficient way.

- Within the marginalized Denoising Auto-Encoder, we fuse infinite ensemble members into a consensus one by adding drop-out noises, which maximizes the capacity of ensemble clustering. Two versions of IEC are proposed with different deep representations.

- Extensive experimental results on numerous real-world data sets with different levels of features demonstrate IEC has obvious advantages on effectiveness and efficiency compared with the state-of-the-art deep clustering and ensemble clustering methods, and IEC is a promising tool for large-scale image clustering.

- The real-world pan-omics gene expression analysis application illustrates the effectiveness of IEC to handle multi-view and incomplete data clustering.

4.1 Problem Definition

Although ensemble clustering can be roughly generalized into two categories, based on co-association matrix or utility function, Liu et al. [48] built a connection between the methods based on co-association matrix and utility functions and pointed out the co-association matrix plays a determinative role in the success of ensemble clustering. Thus, here we focus on the methods based on co-association matrix. Next, we introduce the impact of the number of basic partitions by the following theorem.

**Theorem 4.1.1 (Stableness [82])** For any $\epsilon > 0$, there exists a matrix $S_0$, such that

$$
\lim_{r \to \infty} P(||S - S_0||_F^2 > \epsilon) = 0,
$$

where $|| \cdot ||_F^2$ denotes the Frobenius norm.

From the above theorem, we have the conclusion that although basic partitions might be greatly different from each other due to different generation strategies, the normalized co-association matrix becomes stable with the increase of the number of basic partitions $r$. From our previous experimental results in Chapter 2, it is easy to observe that with the increasing number of basic partitions, the performance of ensemble clustering goes up and becomes stable. However, the best number of basic partitions for a given data set is difficult to set. Too few basic partitions can not exert
CHAPTER 4. INFINITE ENSEMBLE CLUSTERING

the capacity of ensemble clustering, while too many basic partitions lead to unnecessary computational resource waste. Therefore, fusing infinite basic partition is addressed in this paper, instead of answering the best number of basic partitions for a given data set. According to Theorem 4.1.1 we expect to fuse infinite basic partitions to maximize the capacity of ensemble clustering. Since we cannot generate infinite basic partitions, how to obtain a stable co-association matrix $S$ and calculate $H^*$ in an efficient way is highly needed, which is also one of our motivations. In Section 4.2 we employ mDAE to equivalently obtain the “infinite” basic partitions and achieve the expectation of co-association matrix. Deep structure and clustering techniques are powerful tools for computer vision and data mining applications. Especially, ensemble clustering attracts a lot of attention due to its appealing performance. However, these two powerful tools are usually used separately. Notice that the performance of ensemble clustering heavily depends on the basic partitions. As mentioned before, co-association matrix $S$ is the key factor for the ensemble clustering and with the increase of basic partitions, the co-association matrix becomes stable. According to Theorem 4.1.1 the capability of ensemble clustering goes to the upper bound with the number of basic partitions $r \rightarrow \infty$, Then we aim to seamlessly integrate deep concept and ensemble clustering in a one-step framework: Can we fuse infinite basic partitions for ensemble clustering in a deep structure?

The problem is very straightforward, but it is quite difficult. The challenges of the problem lie in three folds:

- How to generate infinite basic partitions?
- How to seamlessly integrate the deep concept within ensemble clustering framework?
- How to solve it in a highly efficient way?

4.2 Infinite Ensemble Clustering

Here we first uncover the connection between ensemble clustering and auto-encoder. Next, marginalized Denoising Auto-Encoder is applied for the expectation of co-association matrix, and finally we propose our method and give the corresponding analysis.

4.2.1 From Ensemble Clustering to Auto-encoder

It seems that there exists no explicit relationship between ensemble clustering and auto-encoder due to their respective tasks. The aim of ensemble clustering is to find a cluster structure
CHAPTER 4. INFINITE ENSEMBLE CLUSTERING

based on basic partitions, while auto-encoder is usually used for better feature generation. Actually auto-encoder can be regarded as an optimization method for minimizing the loss function as well.

Recalling that the goal of ensemble clustering is to find a single partition which agrees the basic ones as much as possible, we can understand it in the opposite way that the consensus partition has the minimum loss to present all the basic ones. After we summarize all the basic partitions into the co-association matrix $S$, spectral clustering or some other graph partition algorithms can be conducted on the co-association matrix to obtain the final consensus result. Taking spectral clustering as an example, we aim to find a $n \times K$ low-dimensional space to represent the original input. Each column of low-dimensional matrix is a base for spanning the space. Then K-means can be run on that for the final partition. Similarly, the function of auto-encoder is also to learn a hidden representation with $d$ dimensions with “carrying” as much as possible information with the input, where $d$ is a user pre-defined parameter. Therefore, to some extent spectral clustering and auto-encoder have the similar function to learn new representations according to minimizing certain objective function; the difference is that in spectral clustering, the dimension of new representation is $K$, while auto-encoder produces $d$ dimensions. From this view, auto-encoder is more flexible than spectral clustering.

Therefore, we have another interpretation of auto-encoder, which not only can generate robust features, but also can be regarded as an optimization method for minimizing the loss function. By this means, we can feed the co-association matrix into auto-encoder to get the new representation, which has the similar function with spectral clustering, and run K-means on that to obtain the consensus clustering. For the efficiency issue, it is not a good choice to use auto-encoder on the ensemble clustering task due to the large space complexity of co-association matrix $O(n^2)$. We will address this issue in the next subsection.

4.2.2 The Expectation of Co-Association Matrix

According to Theorem 4.1.1 with the number of basic partitions going to infinity, the co-association matrix becomes stable. Before answering how to generate infinite ensemble members, we first solve how to increase the number of basic partitions given the limited ones. The naive way is to apply some generation strategy on the original data to produce more ensemble members. The disadvantages lie in two folds: (1) time consuming, (2) sometimes we only have the basic partitions, and the original data are not accessible. Therefore, without the original data, producing more basic partitions with the limited one is like a clone problem. However, simply duplicating the ensemble members does not work. Here we make several copies of basic partitions and corrupt them with
CHAPTER 4. INFINITE ENSEMBLE CLUSTERING

Algorithm 2 The algorithm of Infinite Ensemble Clustering

Input: \( H^{(1)}, \cdots, H^{(r)} ; r \) basic partitions;
\[ l \]: number of layers for mDAE;
\[ p \]: noise level;
\[ K \]: number of clusters.

Output: optimal \( H^* \);
1: Build the binary matrix \( B \);
2: Apply \( l \) layers stacked linear or non-linear mDAE with \( p \) noise level to get the mapping matrix \( W \);
3: Run K-means on \( BW^T \) to get \( H^* \).

erasing some labels in basic partitions to get new ones. By this means, we have extra incomplete basic partitions and Theorem 4.1.1 also holds for incomplete basic partitions.

By this strategy, we just amply the size of ensemble members, which is still far from the infinity. To solve this challenge we use the expectation of co-association matrix instead. Actually, \( S_0 \) is just the expectation of \( S \), which means if we obtain the expectation of co-association matrix as an input for auto-encoder, our goal can be achieved. Since the expectation of co-association matrix cannot be obtained in advance, we intend to calculate it during the optimization.

Inspired by the marginalized Denoising Auto-Encoder \[86\], which involves the expectation of certain noises during the training, we corrupt the basic partitions and marginalize it for the expectation. By adding drop-out noise to basic partitions, some elements are set to be zero, which means some instances are not involved during the basic partition generation. By this means, we can use marginalized Denoising Auto-Encoder to finish the infinite ensemble clustering task. The function \( f \) in auto-encoder can be linear or non-linear. In this chapter, for efficiency issue we use the linear version for mDAE \[86\] since it has a closed-form formulation.

4.2.3 Linear version of IEC

So far, we solve the infinite ensemble clustering problem with marginalized Denoising Auto-Encoder. Before conducting experiments, we notice that the input of mDAE should be the instances with independent and identically distribution; however, the co-association matrix can be regarded as a graph, which disobeys this assumption. To solve this problem, we introduce a binary matrix \( B \).
CHAPTER 4. INFINITE ENSEMBLE CLUSTERING

Let \( \mathbf{B} = \{b(x)\} \) be a binary data set derived from the set of \( r \) basic partitions \( \mathcal{H} \) as follows:

\[
b(x) = \langle b(x)_1, \cdots, b(x)_r \rangle,
b(x)_i = \langle b(x)_{i1}, \cdots, b(x)_{iK_i} \rangle,
b(x)_{ij} = \begin{cases} 1, & \text{if } \mathbf{H}^{(i)}(x) = j \\ 0, & \text{otherwise} \end{cases}
\]

We can see that the binary matrix \( \mathbf{B} \) is a \( n \times d \) matrix, where \( d = \sum_{i=1}^{r} K_i \). It concatenates all the basic partitions with 1-of-\( K_i \) coding, where \( K_i \) is the cluster number in the basic partition \( \mathbf{H}^{(i)} \). With the binary matrix \( \mathbf{B} \), we have \( \mathbf{B} \mathbf{B}^T = \mathbf{S} \). It indicates that the binary matrix \( \mathbf{B} \) has the same information with the co-association matrix \( \mathbf{S} \). Since \( \mathbf{B} \) obeys the independent and identically distribution, we can put the binary matrix as input for marginalized Denoising Auto-Encoder.

For linear version of IEC, the corresponding mapping for \( \mathbf{W} \) between input and hidden representations is in closed form [86]:

\[
\mathbf{W} = \mathbb{E}[\mathbf{P}][\mathbb{E}[\mathbf{Q}]]^{-1},
\]

where \( \mathbf{P} = \mathbf{B} \mathbf{B}^T = \mathbf{S} \) and \( \mathbf{Q} = \mathbf{B}^T \mathbf{B} = \Sigma \). We add the constant 1 at the last column of \( \mathbf{B} \) and corrupt it with \( p \) level drop-out noise. Let \( \mathbf{q} = [1 - p, \cdots, 1 - p, 1] \in \mathbb{R}^{d+1} \), we have \( \mathbb{E}[\mathbf{P}]_{ij} = \Sigma_{ij} \mathbf{q}_j \) and \( \mathbb{E}[\mathbf{Q}]_{ij} = \Sigma_{ij} \mathbf{q}_i \tau(i, j, \mathbf{q}_j) \). Here \( \tau(i, j, \mathbf{q}_j) \) returns 1 with \( i = j \), and returns \( \mathbf{q}_j \) with \( i \neq j \).

After getting the mapping matrix, \( \mathbf{B} \mathbf{W}^T \) is used as the new representation. By this means, we can recursively apply marginalized Denoising Auto-Encoder to obtain deep hidden representations. Finally, K-means is called to run on the hidden representations for the consensus partition. Since only \( r \) elements are non-zeros in each row of \( \mathbf{B} \), it is very efficient to calculate \( \Sigma \). Moreover, \( \mathbb{E}[\mathbf{P}] \) and \( \mathbb{E}[\mathbf{Q}] \) are both \( (d+1) \times (d+1) \) matrixes. Finally, K-means is conducted on all the hidden representations. Therefore, our total time complexity is \( O(ld^3 + IKnl^d) \), where \( l \) is the number of layers of mDAE, \( I \) is the iteration number in K-means, \( K \) is the cluster number, and \( d = \sum_{i=1}^{r} K_i \ll n \). This indicates our algorithm is linear to \( n \), which can be applied for large-scale clustering. Since K-means is the core technique in IEC, the convergence is guaranteed.

4.2.4 Non-Linear version of IEC

For the non-linear version IEC, we follow the non-linear mDAE with second-order expansion and approximation [85] and have the following objective function:

\[
\ell(x, f(\mu x)) + \frac{1}{2} \sum_{d=1}^{D} \sigma_{xd}^2 \sum_{h=1}^{D_h} \left( \frac{\partial^2 \ell}{\partial z_d^2} \frac{\partial z_d}{\partial x_d} \right)^2,
\]

56
CHAPTER 4. INFINITE ENSEMBLE CLUSTERING

Table 4.1: Experimental Data Sets

<table>
<thead>
<tr>
<th>Data set</th>
<th>Type</th>
<th>Feature</th>
<th>#Instance</th>
<th>#Feature</th>
<th>#Class</th>
<th>#MinClass</th>
<th>#MaxClass</th>
<th>CV</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Letter</td>
<td>character</td>
<td>low-level</td>
<td>20000</td>
<td>16</td>
<td>26</td>
<td>734</td>
<td>813</td>
<td>0.0301</td>
<td>0.9738</td>
</tr>
<tr>
<td>MNIST</td>
<td>digit</td>
<td>low-level</td>
<td>70000</td>
<td>784</td>
<td>10</td>
<td>6313</td>
<td>7877</td>
<td>0.0570</td>
<td>0.1914</td>
</tr>
<tr>
<td>COIL100</td>
<td>object</td>
<td>middle-level</td>
<td>7200</td>
<td>1024</td>
<td>100</td>
<td>72</td>
<td>72</td>
<td>0.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>Amazon</td>
<td>object</td>
<td>middle-level</td>
<td>958</td>
<td>800</td>
<td>10</td>
<td>82</td>
<td>100</td>
<td>0.0592</td>
<td>0.1215</td>
</tr>
<tr>
<td>Caltech</td>
<td>object</td>
<td>middle-level</td>
<td>1123</td>
<td>800</td>
<td>10</td>
<td>85</td>
<td>151</td>
<td>0.2087</td>
<td>0.1638</td>
</tr>
<tr>
<td>DSLr</td>
<td>object</td>
<td>middle-level</td>
<td>157</td>
<td>800</td>
<td>10</td>
<td>8</td>
<td>24</td>
<td>0.3857</td>
<td>0.1369</td>
</tr>
<tr>
<td>Webcam</td>
<td>object</td>
<td>middle-level</td>
<td>295</td>
<td>800</td>
<td>10</td>
<td>21</td>
<td>43</td>
<td>0.1879</td>
<td>0.1289</td>
</tr>
<tr>
<td>ORL</td>
<td>face</td>
<td>middle-level</td>
<td>400</td>
<td>1024</td>
<td>40</td>
<td>10</td>
<td>10</td>
<td>0.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>USPS</td>
<td>digit</td>
<td>middle-level</td>
<td>9298</td>
<td>256</td>
<td>10</td>
<td>708</td>
<td>1553</td>
<td>0.2903</td>
<td>1.0000</td>
</tr>
<tr>
<td>Caltech101</td>
<td>object</td>
<td>high-level</td>
<td>1415</td>
<td>4096</td>
<td>5</td>
<td>67</td>
<td>870</td>
<td>1.1801</td>
<td>1.0000</td>
</tr>
<tr>
<td>ImageNet</td>
<td>object</td>
<td>high-level</td>
<td>7341</td>
<td>4096</td>
<td>5</td>
<td>910</td>
<td>2126</td>
<td>0.3072</td>
<td>1.0000</td>
</tr>
<tr>
<td>Sun09</td>
<td>object</td>
<td>high-level</td>
<td>3238</td>
<td>4096</td>
<td>5</td>
<td>20</td>
<td>1264</td>
<td>0.8970</td>
<td>1.0000</td>
</tr>
<tr>
<td>VOC2007</td>
<td>object</td>
<td>high-level</td>
<td>3376</td>
<td>4096</td>
<td>5</td>
<td>330</td>
<td>1499</td>
<td>0.7121</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

where \( l \) is the loss function in auto-encoder, \( \mu_x = x \) is the mean of \( x \), \( \sigma^2_{x_d} = x^2_p/(1-p) \) is the variance of \( x \) in \( d \)-th dimension with the noise level \( p \), \( f \) is the sigmoid function, \( D \) and \( D_h \) are the dimensions of the input and hidden layers, respectively. The detailed understanding about the non-linear objective function can be found in Ref [85]. A well-known framework Theano[1] is applied for the non-linear mDAE optimization.

Similarity, we feed the binary matrix \( B \) into the non-linear version of mDAE for the mapping function \( W \) and calculate the new presentation for clustering. The algorithm is summarized in Algorithm 2.

4.3 Experimental Results

In this section, we first introduce the experimental settings, then showcase the effectiveness and efficiency of IEC compared with the state-of-the-art deep clustering and ensemble clustering methods. Finally, some impact factors of IEC are thoroughly explored.

4.3.1 Experimental Settings

Data Sets. Thirteen real-world image data sets with true cluster labels are used for experiments. Table 4.1 shows their important characteristics, where #MinClass, #MaxClass, CV and Density denote the instance number of the smallest and biggest clusters, coefficient of variation

CHAPTER 4. INFINITE ENSEMBLE CLUSTERING

Figure 4.2: Sample Images. (a) MNIST is a 0-9 digits data sets in grey level, (b) ORL is an object data set with 100 categories, (c) ORL contains faces of 40 people with different poses and (d) Sun09 is an object data set with different types of cars.

A statistic that characterizes the degree of class imbalance, and the ratio of non-zeros elements, respectively. In order to demonstrate the effectiveness of our IEC, we select the data sets with different levels of features, such as pixel, Surf and deep learning features. The first two are characters and digits data sets, the middle ones are the objects and digits data sets and the last four data sets are with the deep learning features. In addition, these data sets contain different types of images, such as digits, characters, objects. Figure 4.2 shows some samples of these data sets.

**Comparative algorithms.** To validate the effectiveness of the IEC, we compare it with several state-of-the-art methods in terms of deep clustering methods and ensemble clustering methods. MAEC \[86\] applies mDAE to get new representations and runs K-means on it to get the partition. Here MAEC1 uses the orginal features as the input, MAEC2 uses the Laplace graph as the input. GEncoder \[81\] is short for GraphEncoder, which feeds the Laplace graph into the sparse auto-encoder to get new representations. DLC \[79\] jointly learns the feature transform function and discriminative codings in a deep mDAE structure. GCC \[1\] is a general concept of three benchmark ensemble clustering algorithms based on graph: CSPA, HGPA and MCLA, and returns the best result. HCC \[6\] is an agglomerative hierarchical clustering algorithm based on the co-association matrix. KCC \[49\]
### Table 4.2: Clustering Performance of Different Algorithms Measured by Accuracy

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>Baseline</th>
<th>Deep Clustering Method</th>
<th>Ensemble Clustering Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K-means</td>
<td>MAEC1</td>
<td>MAEC2</td>
</tr>
<tr>
<td>letter</td>
<td>0.2485</td>
<td>0.1163</td>
<td>N/A</td>
</tr>
<tr>
<td>MNIST</td>
<td>0.4493</td>
<td>0.3757</td>
<td>N/A</td>
</tr>
<tr>
<td>COIL100</td>
<td>0.5056</td>
<td>0.0124</td>
<td>0.5206</td>
</tr>
<tr>
<td>Amazon</td>
<td>0.3309</td>
<td>0.4395</td>
<td>0.2443</td>
</tr>
<tr>
<td>Caltech</td>
<td>0.2457</td>
<td>0.2787</td>
<td>0.2102</td>
</tr>
<tr>
<td>Dslr</td>
<td>0.3631</td>
<td>0.4140</td>
<td>0.3185</td>
</tr>
<tr>
<td>Webcam</td>
<td>0.3932</td>
<td>0.5085</td>
<td>0.3220</td>
</tr>
<tr>
<td>ORL</td>
<td>0.5475</td>
<td>0.0450</td>
<td>0.3675</td>
</tr>
<tr>
<td>USPS</td>
<td>0.6222</td>
<td>0.6290</td>
<td>0.4066</td>
</tr>
<tr>
<td>Caltech101</td>
<td>0.6898</td>
<td>0.4311</td>
<td>0.5060</td>
</tr>
<tr>
<td>ImageNet</td>
<td>0.6675</td>
<td>0.6601</td>
<td>0.3483</td>
</tr>
<tr>
<td>Sun09</td>
<td>0.4360</td>
<td>0.4750</td>
<td>0.3696</td>
</tr>
<tr>
<td>VOC2007</td>
<td>0.4565</td>
<td>0.4138</td>
<td>0.3874</td>
</tr>
</tbody>
</table>

is a K-means-based consensus clustering which transfers the ensemble clustering into a K-means optimization problem. SEC [48] employs spectral clustering on co-association matrix and solves it by weighted K-means.

In the ensemble clustering framework, we employ Random Parameter Selection (RPS) strategy to generate basic partitions. Generally speaking, $k$-means is conducted on all features with different numbers of clusters, varying from $K$ to $2K$. To show the best performance of the comparative algorithms, 100 basic partitions via RPS are produced for boosting the comparative methods. Note that we set 5 layers in our linear model and 1 layer for the non-linear model, and set the dimension of the hidden layers as the same with the one of input layer. For all clustering methods, we set $K$ to be the true cluster number for fair comparison.

**Validation metric.** Since the label information is available to these data sets, here we use two external metrics accuracy and Normalized Mutual Information (NMI) to measure the performance. Note that accuracy and NMI are both positive measurements, which means the larger, the better.

**Environment.** All the experiments except the non-linear IEC were run on a Windows standard platform of 64-bit edition, which has two Intel Core i7 3.4GHz CPUs and 32GB RAM. The non-linear IEC was conducted on a Ubuntu 14.04 of 64-bit edition with a NVIDIA TITAN X GPU.
Table 4.3: Clustering Performance of Different Algorithms Measured by NMI

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>Baseline</th>
<th>Deep Clustering Method</th>
<th>Ensemble Clustering Method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>K-means</td>
<td>MAEC</td>
<td>MAEC2</td>
</tr>
<tr>
<td>letter</td>
<td>0.3446</td>
<td>0.1946</td>
<td>N/A</td>
</tr>
<tr>
<td>MNIST</td>
<td>0.4542</td>
<td>0.3086</td>
<td>N/A</td>
</tr>
<tr>
<td>COIL100</td>
<td>0.7719</td>
<td>0.0769</td>
<td>0.7794</td>
</tr>
<tr>
<td>Amazon</td>
<td>0.3057</td>
<td>0.3588</td>
<td>0.1982</td>
</tr>
<tr>
<td>Caltech</td>
<td>0.2043</td>
<td>0.1862</td>
<td>0.1352</td>
</tr>
<tr>
<td>Dslr</td>
<td>0.3766</td>
<td>0.4599</td>
<td>0.2900</td>
</tr>
<tr>
<td>Webcam</td>
<td>0.4242</td>
<td>0.5269</td>
<td>0.2316</td>
</tr>
<tr>
<td>ORL</td>
<td>0.7651</td>
<td>0.2302</td>
<td>0.6268</td>
</tr>
<tr>
<td>USPS</td>
<td>0.6049</td>
<td>0.4722</td>
<td>0.4408</td>
</tr>
<tr>
<td>Caltech101</td>
<td>0.7188</td>
<td>0.4980</td>
<td>0.5200</td>
</tr>
<tr>
<td>ImageNet</td>
<td>0.4287</td>
<td>0.4827</td>
<td>0.1556</td>
</tr>
<tr>
<td>Sun09</td>
<td>0.2014</td>
<td>0.2787</td>
<td>0.0576</td>
</tr>
<tr>
<td>VOC2007</td>
<td>0.2697</td>
<td>0.2653</td>
<td>0.1118</td>
</tr>
</tbody>
</table>

4.3.2 Clustering Performance

Table 4.2 and 4.3 show the clustering performance of different algorithms in terms of accuracy and NMI. The best results are highlighted in bold font. “N/A” denotes there is no result due to out of memory. As can be seen from the tables, three observations are very clear. (1) In the deep clustering method, MAEC1 performs the best and the worst on Amazon and COIL100, respectively; on the contrary, MAEC2 gets reasonable result on COIL100 but low quality on Amazon. Although we try our best to tune the number of neurons in the hidden layers, GEncoder suffers from the worst performance in all the comparative methods, even worse than K-means. The high computational cost prohibits MAEC2 and GEncoder from handling large-scale data sets. Since clustering belongs to the unsupervised learning, only relying on deep structure makes little effect to improve the performance. Instead DLC jointly learns the feature transform function and discriminative codings in a deep structure, which has the satisfactory results. (2) In most cases, ensemble clustering is superior to the baseline method, even better than deep clustering methods. The improvement is obvious when applying ensemble clustering methods on the data sets with high-level features, since high-level features have more structural information. However, ensemble methods do not work well on SUN09. One of the reasons might be the unbalanced class structure, which prevents the basic clustering algorithm K-means from uncovering the true structure and further harms the performance of ensemble methods. (3) Our method IEC gets the best results on most of 13 data sets. It is worthy to note that the improvements are over nearly 8%, 8% or 22% on Dslr, USPS and Caltech101, respectively,
CHAPTER 4. INFINITE ENSEMBLE CLUSTERING

which are rare in clustering field. Usually the performance of ensemble clustering goes up with the increase the number of basic partitions. In order to show the best performance of the comparative ensemble clustering methods, we use 100 basic partitions. Here we can see that there still exists large space to improve via infinite ensemble members.

For efficiency, to make fair comparisons here we only report the execution time of ensemble clustering methods. Although additional time is needed for generating basic partitions, \textit{k-means} and parallel computation make it quite efficient. Table 4.4 shows the average time of ten runs via these methods. GCC runs three methods on small data sets but runs two methods on large data sets, and HCC runs fast on data sets containing few instances but struggles as the number of instances increases due to its $O(n^3)$ time complexity. KCC, SEC and IEC are all K-means-based methods, which are much faster than other ensemble methods. Since our method only applies mDAE on basic

Figure 4.3: Running time of linear IEC with different layers and instances.

Table 4.4: Execution time of different ensemble clustering methods by second

<table>
<thead>
<tr>
<th>Data sets</th>
<th>GCC</th>
<th>HCC</th>
<th>KCC</th>
<th>SEC</th>
<th>IEC (5 layers)</th>
</tr>
</thead>
<tbody>
<tr>
<td>letter</td>
<td>383.89</td>
<td>1717.88</td>
<td>11.39</td>
<td>8.35</td>
<td>55.46</td>
</tr>
<tr>
<td>MNIST</td>
<td>112.44</td>
<td>19937.69</td>
<td>11.98</td>
<td>3.79</td>
<td>51.55</td>
</tr>
<tr>
<td>COIL100</td>
<td>21.27</td>
<td>170.02</td>
<td>4.99</td>
<td>3.09</td>
<td>14.93</td>
</tr>
<tr>
<td>Amazon</td>
<td>3.93</td>
<td>1.61</td>
<td>0.17</td>
<td>0.08</td>
<td>1.21</td>
</tr>
<tr>
<td>Caltech</td>
<td>3.55</td>
<td>2.12</td>
<td>0.23</td>
<td>0.11</td>
<td>1.43</td>
</tr>
<tr>
<td>Dslr</td>
<td>2.27</td>
<td>0.09</td>
<td>0.04</td>
<td>0.06</td>
<td>0.70</td>
</tr>
<tr>
<td>Webcam</td>
<td>2.09</td>
<td>0.14</td>
<td>0.04</td>
<td>0.05</td>
<td>0.90</td>
</tr>
<tr>
<td>ORL</td>
<td>6.81</td>
<td>0.04</td>
<td>0.21</td>
<td>0.21</td>
<td>14.11</td>
</tr>
<tr>
<td>USPS</td>
<td>7.66</td>
<td>160.41</td>
<td>1.73</td>
<td>0.53</td>
<td>5.48</td>
</tr>
<tr>
<td>Caltech101</td>
<td>1.21</td>
<td>1.68</td>
<td>0.15</td>
<td>0.09</td>
<td>0.53</td>
</tr>
<tr>
<td>ImageNet</td>
<td>3.83</td>
<td>52.47</td>
<td>1.40</td>
<td>0.32</td>
<td>1.76</td>
</tr>
<tr>
<td>Sun09</td>
<td>2.36</td>
<td>10.01</td>
<td>0.33</td>
<td>0.13</td>
<td>0.82</td>
</tr>
<tr>
<td>VOC2007</td>
<td>2.05</td>
<td>10.97</td>
<td>0.32</td>
<td>0.16</td>
<td>0.82</td>
</tr>
</tbody>
</table>
CHAPTER 4. INFINITE ENSEMBLE CLUSTERING

partitions which has the closed-form solution and then runs K-means on the new representations, therefore IEC is suitable for large-scale image clustering. Moreover, Figure 4.3 shows the running time on MNIST and letter with different number of layers and instances. We can see that the running time is linear to the layer number and instant number, which verifies the high efficiency of IEC. Therefore, if we only use one layer in IEC, the execution time is similar to KCC and SEC.

In the end of this subsection, we compare the clustering performance of linear and non-linear IEC in Figure 4.4. Here we employ 5-layer linear model and one-layer non-linear model. From Figure 4.4, we can see that the non-linear model has 2%-6% improvements on Webcam and ORL over the linear one in terms of accuracy. However, the non-linear model is an approximate calculation while linear model has closed-form representation. Besides, the non-linear model takes long time
to train even with the GPU accelerator. Taking the effectiveness and efficiency into comprehensive consideration, we choose the linear version of IEC as our default model for further analysis.

### 4.3.3 Inside IEC: Factor Exploration

Next we thoroughly explore the impact factors of IEC in terms of the number of layers, the generation strategy of basic partitions, the number of basic partitions, and the noise level, respectively.

**Number of layers.** Since stacked marginalized Denoising Auto-Encoder is used to fuse infinite ensemble members, here we explore the impact of the number of layers. As can be seen in Figure 4.5(a), the performance of IEC goes slightly up with the increase of layers. Except that the second layer has large improvements over the first layer on *Caltech101*, IEC demonstrates the stable results on different layers, because only one-layer marginalized Denoising Auto-Encoder calculates the expectation of co-association matrix. Usually the deep representation is successful in many applications on computer vision, here the default value of the number of layers is set to be 5.

**Generation strategy of basic partitions.** So far we rely solely on Random Parameter
CHAPTER 4. INFINITE ENSEMBLE CLUSTERING

Selection (RPS) to generate basic partitions, with the number of clusters varying in \([K, 2K]\). In the following, we demonstrate whether the generation strategy will impact the performance of IEC.

Here Random Feature Selection (RFS) is proposed as a comparison, which still uses \(k\)-means as the basic clustering algorithm with random selecting 50% original features to obtain 100 basic partitions. Figure 4.5(b) demonstrates the performance of KCC and IEC via RPS and RFS on 5 data sets. As we can see that, IEC exceeds KCC in most cases of RPS and RFS. When we take a close look, the performance of IEC via RPS and RFS is almost the same, while KCC produces large gaps between RPS and RFS on Caltech101 and Sun09 (See the ellipses). This indicates that although the generation of basic partitions is of high importance to the success of ensemble clustering, we can take use of infinite ensemble clustering to alleviate the impact.

**Number of basic partitions.** The key problem of this chapter is to use limited basic partitions to achieve the goal of fusing infinite ensemble members. Here we discuss the impact of the number of basic partitions to ensemble clustering. Figure 4.5(c) shows the performance of 4 ensemble clustering methods on USPS. Generally speaking, the performance of HCC, KCC and GCC goes up with the increase of the number of basic partitions and becomes stable when enough basic partitions are given, which is consistent with Theorem 4.1.1. Moreover, we can see that the co-association matrices have clear block-structure with more basic partitions in Figure 4.6. It is worthy to note that IEC enjoys the high performance even with 5 ensemble members. It is worthy to note that for large-scale data sets, generating basic partition suffers from high time complexity even with ensemble process. Thus, it is appealing that IEC uses limited basic partitions and achieves the high performance, which is suitable for tons of image clustering.

**Noise level.** The core idea of this chapter is to obtain the expectation of co-association matrix via adding the drop-out noise. Figure 4.5(d) shows the results of IEC with different noise level on four data sets. As can be seen that, the performance of IEC is quite stable even to 0.5 noise level.
4.4 Application on Pan-omics Gene Expression Analysis

With the rapid development of techniques, it becomes much more easier to collect diverse and rich molecular data types from genome to transcriptome, proteome, and epigenome [93, 94]. The pan-omics gene expressions, which is also known as multi-view data, provide great opportunities to characterize human pathologies and disease subtypes, identify driver genes and pathways, and nominate drug targets for precision medicine [95, 96]. Clustering, an unsupervised exploratory analysis, has been widely used for patient stratification or disease subtyping [97, 98]. To fully demonstrate the effectiveness of IEC in real-world applications, here we employ IEC for pan-omics gene analysis. In the following, we introduce the gene expression data sets and the experimental setting, evaluate the performance of different clustering methods by survival analyses and finally apply IEC for the missing pan-omics gene expression analysis.

4.4.1 Experimental Setting

Data Sets. Thirteen pan-omics gene expression data sets with survival information from TCGA are used for evaluating the performance of patient stratification. These data sets denote the
CHAPTER 4. INFINITE ENSEMBLE CLUSTERING

gene expression of the patients with 13 major cancer types and each data set contains 4 different types of molecular data, including protein expression, microRNA (miRNA) expression, mRNA expression (RNA-seq V2) and somatic copy number alterations (SCNAs). These cancer types include bladder urothelial carcinoma (BLCA), breast cancer carcinoma (BRCA), colon adenocarcinoma (COAD), head and neck squamous cell carcinoma (HNSC), kidney renal clear cell carcinoma (KIRC), acute myeloid leukemia (LAML), brain lower grade glioma (LGG), lung adenocarcinoma (LUAD), lung squamous cell carcinoma (LUSC), ovarian serous cystadenocarcinoma (OV), prostate adenocarcinoma (PRAD), skin cutaneous melanoma (SKCM), thyroid carcinoma (THCA), and uterine corpus endometrial carcinoma (UCEC). Table 4.5 shows some key characteristic of 13 real-world datasets from TCGA. These four types of molecular data have different dimensions. For example, the protein expression has 190 dimensions, miRNA expression has 1,046 dimensions, mRNA expression has 20,531 dimensions and SCNA has 24,952 dimensions. It is also worthy to note that the numbers of subjects on different molecular types on each data set are different due to the missing data or device failure.

Comparative algorithms. Since we focus on the gene expression analysis, some widely used clustering methods in biological domain are chosen for comparison in terms of traditional clustering and ensemble clustering methods. Agglomerative hierarchical clustering, K-means (KM) and spectral clustering (SC) are baseline methods. Here agglomerative hierarchical clustering with the group-linkage, single-linkage and complete-linkage denotes as AL, SL and CL. LCE \([99]\) is a link-based cluster ensemble method, which accesses the similarity between two clusters, builds refined co-association matrix, and applies spectral clustering for the final partition. ARSR \([100]\) is short for Approximated Sim-Rank Similarity (ASRS) matrix, which is based on a bipartite graph representation of the cluster ensemble in which vertices represent both clusters and data points and edges connect data points to the clusters to which they belong.

Similar to the setting in Section 4.3, we still use Random Parameter Selection (RPS) strategy with the cluster numbers varying from \(K\) to \(2K\) to generate 100 basic partitions for the ensemble clustering methods LCE, ARSR and IEC. And for all clustering methods, we set \(K\) to be the true cluster number for fair comparison.

Validation metric. For these 13 real-world molecular data without label information, we employ survival analyses to evaluate the performance of different clustering methods. Survival analysis considers the expected duration of time until one or more events happen, such as death, disease occurrence, disease recurrence, recovery, or other experience of interest \([101]\). Based on the partition obtained by different clustering methods, we divide the objects or patients into several different groups. Then survival analyses are conducted to calculate whether these groups have
significant differences by log-rank test.

The log-rank test is a hypothesis test to compare the survival distributions of two or more groups. The null hypothesis is that every group has the same or similar survival function. The expected number of subjects surviving at each time point in each group is adjusted for the number of subjects at risk in the groups at each event time. The log-rank test determines if the observed number of events in each group is significantly different from the expected number. The formal test is based on a chi-squared statistic. The log-rank statistic has a chi-squared distribution with one degree of freedom, and the p-value is calculated using the chi-squared distribution. When the p-value is smaller
than 0.05, it typically indicates that those groups differ significantly in survival times. Here survival library in R package\footnote{https://cran.r-project.org/web/packages/survival/index.html} is used for the log-rank test.

**Environment.** All the experiments were run on a Windows standard platform of 64-bit edition, which has two Intel Core i7 3.4GHz CPUs and 32GB RAM.

### 4.4.2 One-omics Gene Expression Evaluation

Since these 13 data sets have different numbers of instances within four different types, we first evaluate these widely used clustering methods in biological domain and IEC in the one-omics setting. That means that we treat these 13 data sets with four modular types as 52 independent data sets, and then run clustering methods and evaluate the performance of survival analysis by p-value. For the ensemble methods, LCE, ARSR and IEC, RPS strategy is employed to generate 100 basic partitions. And for all clustering methods, we set $K$ to be the true cluster number for fair comparison.

Figure 4.7 shows the survival analysis performance of different clustering methods on one-omics setting, where colors denote the $-\log(p\text{-value})$ of the survival analysis. For better comparison, we set $-\log(0.05)$ as the white color so that the warm colors (yellow, orange and red) mean the pass of hypothesis test and the cold colors (blue) mean the failure of hypothesis test. From this figure, we have three observations. (1) Generally speaking, traditional clustering methods, such as agglomerative hierarchical clustering, K-means and spectral clustering deliver poor performance, especially AL has no pass on the miRNA modular data. Compared with these traditional clustering methods, ensemble methods fuse several diverse basic partitions and enjoy more passes on these data sets. (2) IEC shows the obvious advantages over other competitive methods with more bright
CHAPTER 4. INFINITE ENSEMBLE CLUSTERING

Figure 4.9: Execution time in logarithm scale of different ensemble clustering methods on 13 cancer data sets with 4 different molecular types.

area and more passes of hypothesis tests. On KIRC data set with protein expression, BLCA and UCEC data sets with miRNA expression, BLCA, OV, SCKM and THCA data sets with SCNA, only IEC passes the hypothesis tests. Figure 4.8 shows the number of passed hypothesis tests of these clustering methods on four different modular types. On these 52 independent data sets, IEC has 38 passes hypothesis tests with the passing rate over 73.0%, while the second best method only has the 32.7% passing rate. The benefits of IEC lie in two aspects. One is that IEC is an ensemble clustering method, which incorporates several basic partitions in a high-level fusion fashion; the other is that the latent infinite partitions make the results resist to noises. (3) Different types of molecular data have different capacities to uncover the cluster structure for survival analysis. For example, most of methods pass the hypothesis tests on mRNA, while few of them pass the hypothesis tests on SCNA. For a certain data set or cancer, we cannot pre-know what is the best molecular data type for passing the hypothesis test of survival analysis. In light of this, we aim to provide the pan-omics gene expression evaluation in the next subsection.

Figure 4.9 shows the execution time in logarithm scale of LCE, ASRS and IEC on 13 cancer data sets with 4 different molecular types. Since IEC enjoys the roughly linear time complexity to the number of instance, IEC has significant advantages over LCE and ASRS in terms of efficiency. For example, IEC is 2 to 4 times faster than LCE and 20 to 66 times faster than ASRS. This indicates that IEC is a suitable ensemble clustering tool for real-world applications in large-scale.
4.4.3 Pan-omics Gene Expression Evaluation

In this subsection, we continue to evaluate the performance of IEC with missing values. In the pan-omics application, it is quite normal to collect the data with missing values or missing instances. For example, these 13 cancer data sets in Table 4.5 have different numbers of instances in different types. To handle the missing data, a naive way is to remove the instances with missing values so that a smaller complete data set can be achieved. However, this way is a kind of waster since collecting data is very expensive especially in biology domain. Although there exist missing values in the pan-omics gene expression in Table 4.5, we can still employ the IEC to finish the partition.

To achieve this, we generate 25 incomplete basic partitions for each one-omics gene expression by running K-means on incomplete data sets and the missing instances are labelled as zeros. Then IEC is applied to fuse 100 incomplete basic partitions into the consensus one. Figure 4.10 shows the survival analysis of IEC on 13 pan-omics data sets. We can see that by integrating pan-omics gene expression, IEC passes all the hypothesis tests on 13 cancer data sets. Recall that in the one-omics setting, IEC fails the hypothesis tests on some data sets. This indicates that even incomplete pan-omics gene expression is conductive to uncover the meaningful structure. Figure 4.11 shows the survival curves of four cancer data sets by IEC.
4.5 Summary

In this chapter, we proposed a novel ensemble clustering algorithm Infinite Ensemble Clustering (IEC) to fuse infinite basic partitions. Generally speaking, we built a connection between ensemble clustering and auto-encoder, and applied marginalized Denoising Auto-Encoder to fuse infinite incomplete basic partitions. The linear and non-linear versions of IEC were provided. Extensive experiments on 13 data sets with different levels of features demonstrated our method IEC had promising performance over the state-of-the-art deep clustering and ensemble clustering methods; besides, we thoroughly explored the impact factors of IEC in terms of the number of layers, the generation strategy of basic partitions, the number of basic partitions, and the noise level to show the robustness of our method. Finally, we employed 13 pan-omics gene expression cancer data sets to illustrate the effectiveness of IEC in the real-world applications.
Chapter 5

Partition-Level Constraint Clustering

Cluster analysis is a core technique in machine learning and artificial intelligence [102, 103, 104], which aims to partition the objects into different groups that objects in the same group are more similar to each other than to those in other groups. It has been widely used in various domains, such as search engines [105], recommend systems [106] and image segmentation [107]. In light of this, many algorithms have been proposed to thrive this area, such as connectivity-based clustering [108], centroid-based clustering [35] and density-based clustering [109]; however, the results of clustering still exist large gaps with the results of classification. To further improve the performance, constrained clustering comes into being, which incorporates pre-known or side information into the process of clustering.

Since clustering has the property of non-order, the most common constraints are pairwise. Specifically, Must-Link and Cannot-Link constraints represent that two instances should lie in the same cluster or not [110, 111]. At the first thought, it is easy to decide Must-Link or Cannot-Link for pairwise comparison. However, in real-world applications, just given one image of a cat and one image of a dog (See Fig. 5.1), it is difficult to answer whether these two images should be in a cluster or not because no decision rule can be made based on only two images. Without additional objects as references, it is highly risky to determine whether the data set is about cat-and-dog or animals-and-non-animals. Besides, as Ref. [112] reported, large disagreements are often observed among human workers in specifying pairwise constraints; for instance, more than 80% of the pairwise labels obtained from human workers are inconsistent with the ground truth for the Scenes data set [113]. Moreover, it has been widely recognized that the order of constraints also has great impact on the clustering results [114], therefore sometimes more constraints even make a detrimental effect. Although some methods such as soft constraints [115, 116] are put forward to handle these
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

Figure 5.1: The comparison between pairwise constraints and partition level side information. In (a), we cannot decide a Must-Link or Cannot-link only based on two instances; compared (b) with (c), it is more natural to label the instances in well-organised way, such as partition level rather than pairwise constraint.

challenges, the results are still far away from satisfactory.

In response to this, we use partition level side information to address these limitations of pairwise constraints. Partition level side information also called partial labeling means that only a small portion of data is labeled into different clusters. Compared with pairwise constraints, partition level side information has the following benefits: (1) it is more natural to organize the data in a higher level than pairwise comparisons, (2) when human workers label one instance, other instances provide enough information as reference for a good decision, (3) it is immune to the self-contradiction and the order of pairwise constraints. The concept of partition level side information was proposed by [117], which aims to find better initialization centroids and employs the standard K-means to finish the clustering task; since the partition level side information is only used to initialize the centroids without involving it into the process of clustering, this method does not belong to the constrained clustering area. In this chapter, we revisit partition level side information and involve it into the process of clustering to obtain the final solution in a one-step framework. Inspired by the success of ensemble clustering [48], we take the partition level side information as a whole and calculate the similarity between the learnt clustering solution and the given side information. We propose the Partition Level Constrained Clustering (PLCC) framework, which not only captures the intrinsic structure from data, but also agrees with the partition level side information as much as possible. Based on K-means clustering, we derive the objective function and give its corresponding solution via derivation. Further, the above solution can be equivalently transformed into a K-means-like optimization problem with only slight modification on the distance function.
and update rule for centroids. Thus, a roughly linear time complexity can be guaranteed. Moreover, we extend it to handle multiple side information and provide the algorithm of partition level side information for spectral clustering. Extensive experiments on several real-world datasets demonstrate the effectiveness and efficiency of our method compared to pairwise constrained clustering and ensemble clustering, even in the inconsistent cluster number setting, which verify the superiority of partition level side information for the clustering task. Besides, our K-means-based method has high robustness to noisy side information even with 50% noisy side information. And we validate the performance of our method with multiple side information, which makes it a promising candidate for crowdsourcing. Finally, an unsupervised framework called Saliency-Guided Constrained Clustering (SG-PLCC) is put forward for the image cosegmentation task, which demonstrates the effectiveness and flexibility of PLCC in different domains. Our main contributions are highlighted as follows.

- We revisit partition level side information and incorporate it to guide the process of clustering and propose the Partition Level Constrained Clustering framework.
- Within the PLCC framework, we propose a K-means-like algorithm to solve the clustering with partition level side information in a highly efficient way and extend our model to multiple side information and spectral clustering.
- Extensive experiments demonstrate our algorithm not only has promising performance compared to the state-of-the-art methods, but also exhibits high robustness to noisy side information.
- A cosegmentation application with saliency prior is employed to further illustrate the flexibility of PLCC. Although only the raw features are extracted and K-means clustering is conducted, we still achieve promising results compared with several cosegmentation algorithms.

### 5.1 Constrained Clustering

K. Wagstaff and C. Cardie first put forward the concept of constrained clustering via incorporating pairwise constraints (Must-Link and Cannot-Link) into a clustering algorithm and modified COBWEB to finish the partition [110]. Later, COP-K-means, a K-means-based algorithm kept all the constraints satisfied and attempted to assign each instance to its nearest centroid [111]. [119] developed a framework to involve pre-given knowledge into density estimation with Gaussian
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

Mixture Model and presented a closed-form EM procedure and generalized EM procedure for Must-Link and Cannot-Link respectively. These algorithms can be regarded as hard constrained clustering since they do not allow any violation of the constraints in the process of clustering. However, sometimes satisfying all the constraints as well as the order of constraints make the clustering intractable and no solution often can be found.

To overcome such limitation, soft constrained clustering algorithms have been developed to minimize the number of violated constraints. Constrained Vector Quantization Error (CVQE) considered the cost of violating constraints and optimized the cost within the objective function of K-means \[114\]. Further, LCVQE modified CVQE with different computation of violating constraints \[115\]. Metric Pairwise Constrained K-means (MPCK-means) employed the constraints to learn a best Mahalanobis distance metric for clustering \[116\]. Among these K-means-based constrained clustering, \[120\] presented a thorough comparative analysis and found that LCVQE presents better accuracy and violates fewer constraints than CVQE and MPCK-Means. It is worthy to note that an NMF-based method also incorporates the partition level side information for constrained clustering \[121\], which requires that the data points sharing the same label have the same coordinate in the new representation space.

Another category of constrained clustering is to incorporate constraints into spectral clustering, which can be roughly generalized into two groups. The first group directly modifies the Laplacian graph. Kamvar et al. proposed the spectral learning method which set the entry to 1 or 0 according to Must-link and Cannot-link constraints and employed the traditional spectral clustering to obtain the final solution \[122\]. Similarly, Xu et al. used the similar way to modify the graph and applied random walk for clustering \[123\]. Lu et al. propagated the constrains in the affinity matrix \[124\], \[125\] and \[126\] combined the constraint matrix as a regularizer to modify the affinity matrix. The second group modifies the eigenspace instead. \[127\] altered the eigenspace according to the hard or soft constraints. Li et al. enforced constraints by regularizing the spectral embedding \[128\]. Recently, \[129\] proposed a flexible constrained spectral clustering to encode the constraints as part of a constrained optimization problem.

5.2 Problem Formulation

In this section, we first give the definition of partition level side information and uncover the relationship between partition level side information, pairwise constraints and ground truth labels. Then based on partition level side information, we give the problem definition, build the model and
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

derive its corresponding solution; further an equivalent solution is designed by modified K-means in an efficient way. Finally, the model is extended to handle multiple side information.

5.2.1 Partition Level Side Information

Since clustering is an orderless partition, pairwise constraints are employed to further improve the performance of clustering for a long time. Specifically, Must-Link and Cannot-Link constraints represent that two instances should lie in the same cluster or not. Although within the framework of pairwise constraints we avoid answering the mapping relationship among different clusters and at the first thought it is easy to make the Must-Link or Cannot-Link decision for pairwise constraints, such pairwise constraints are illogic in essence. For example (See Figure 5.1), given one pair images of a cat and a dog, it cannot be directly determined whether these two images are in the same cluster or not without external information, such as human knowledge or expert suggestion. Here comes the first question that what is the cluster. The goal of cluster analysis is to find cluster structure. Only after clustering, we can summarize the meaning for each cluster. If we already know the meaning of each cluster, the problem becomes the classification problem, rather than clustering. Given that we do not know the meaning of clusters in advance, it is highly risky to make the pairwise constraints. Someone might argue that experts have their own pre-defined cluster structure, but the matching between pre-defined and true cluster structure also begs questions. Take Fig. 5.1 as an example. For the cat and dog images, users might have different decision rules based on different pre-defined cluster structures, such as animal or non-animal, land, water or flying animal and just cat or dog categories. That is to say, without seeing other instances as references, the decisions we make based on two instances suffer from high risk. More importantly, pairwise constraints disobey the way we make decisions. The data should be organized in a higher level rather than pairwise comparisons. Besides, it is tedious to build a pairwise constraint matrix with only 100 instances. Even though the pairwise constraints matrix is a symmetric matrix and there exists transitivity for Must-Link and Cannot-Link constraints, the size of elements of the pairwise constraints matrix is relatively huge to the number of instances.

To avoid these drawbacks of pairwise constraints, here we leverage a new constraint for clustering, called partition level side information as follows.

Definition 3 (Partition Level Side Information) Given a data set containing \( n \) instances, randomly select a small portion \( p \in (0, 1) \) of the data to label from 1 to \( K \), which is the user-predefined cluster
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

number, then the label information for only small portion of the data is called \( p \)-partition level side information.

Different from pairwise constraints, partition level side information groups the given \( np \) instances as a whole process. Taking other instances as references, it makes more sense to decide the group labels than pairwise constraints. Another benefit is that partition level side information has high consistency, while sometimes pairwise constraints from users might be self-contradictory by transitivity. That is to say, given a \( p \)-partition level side information, we can build an \( np \times np \) pairwise constraints matrix with containing the same information. On the contrary, a \( p \)-partition level side information cannot be derived by several pairwise constraints. In addition, for human beings it is much easier to separate an amount of instances into different groups, which accords with the way of labeling. As above mentioned, partition level side information has obvious advantages over pairwise constraints, which is also a promising candidate for crowd sourcing labeling.

It is also worth illustrating the difference between partition level side information and ground truth. Partition level side information is still an orderless partition. However, if we exchange the labels of ground truth, they become wrong labels. Another point is that partition level side information coming from users might have different cluster numbers, even suffer from noisy and wrong decision makings. Besides partition level side information comes from multi-users, which might differ from each other, while the ground truth is unique. Especially in the labeling task, the partial labeled data might have the fewer cluster number than the one of the whole data. In this case, we cannot transform the constrained clustering problem into the traditional classification problem.

5.2.2 Problem Definition

Based on the Definition 3 of partition level side information, we formalize the problem definition: *How to utilize partition level side information to better conduct clustering?*

This problem is totally new to the clustering area. To solve this problem, we have to handle the following challenges:

- How to fuse partition level side information into the process of clustering?
- What is the best mapping relationship between partition level side information and the cluster structure learned from the data?
- How to handle multi-source partition level side information to guide the generation of clustering?
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

One intuitive way to solve the above problem is to transform the partition level side information into pairwise constraints, then any traditional semi-supervised clustering method can be used to obtain final clustering. However, such solution does not make full use of the advantages of partition level side information. Inspired by the huge success of ensemble clustering, we treat the partition level side information as an integrated one and make the clustering result agree with the given partition level side information as much as possible. Specifically, we calculate disagreement between the clustering result and the given partition level side information from a utility view. Here we take K-means as the basic clustering method and give its corresponding objective function for partition level side information in the following.

5.2.3 Objective Function

Let $X$ be the data matrix with $n$ instances and $m$ features and $S$ be a $np \times K$ side information matrix containing $np$ instances and $K$ clusters, where each row only has one element with value 1 representing the label information and others are all zeros. The objective function of our model is as follows:

$$\min_{H, C, G} \| X - HC \|_F^2 - \lambda U_c(H \otimes S, S)$$

$$s.t. H_{ik} \in \{0, 1\}, \sum_{k=1}^{K} H_{ik} = 1, 1 \leq i \leq n. \quad (5.1)$$

where $H$ is the indicator matrix, $C$ is the centroids matrix, $H \otimes S$ is part of $H$ where the instances are also in the side information $S$, $U_c$ is the well-known categorical utility function \[29\], $\lambda$ is a tradeoff parameter to present the confidence degree of the side information and the constraints make the final solution a hard partition, which means one instance only belongs to one cluster.

The objective function consists of two parts. One is the standard K-means with squared Euclidean distance, the other is a term measuring the disagreement between part of $H$ and the side information $S$. We aim to find a solution $H$, which not only captures the intrinsic structural information from the original data, but also has as little disagreement as possible with the side information $S$.

To solve the optimization problem in Eq. $(5.1)$, we separate the data $X$ and indicator matrix $H$ into two parts, $X_1$ and $X_2$, $H_1$ and $H_2$, according to side information $S$. Therefore, the objective function can be written as:

$$\min_{H_1, H_2} \| X_1 - H_1C \|_F^2 + \| X_2 - H_2C \|_F^2 - \lambda U_c(H_1, S). \quad (5.2)$$
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

Table 5.1: Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Domain</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>( \mathbb{R} )</td>
<td>Number of instances</td>
</tr>
<tr>
<td>( m )</td>
<td>( \mathbb{R} )</td>
<td>Number of features</td>
</tr>
<tr>
<td>( K )</td>
<td>( \mathbb{R} )</td>
<td>Number of clusters</td>
</tr>
<tr>
<td>( p )</td>
<td>( \mathbb{R} )</td>
<td>Percentage of labeled data</td>
</tr>
<tr>
<td>( X )</td>
<td>( \mathbb{R}^{n \times m} )</td>
<td>Data matrix</td>
</tr>
<tr>
<td>( S )</td>
<td>( {0, 1}^{np \times K'} )</td>
<td>Partition level side information</td>
</tr>
<tr>
<td>( H )</td>
<td>( {0, 1}^{n \times K} )</td>
<td>Indicator matrix</td>
</tr>
<tr>
<td>( C )</td>
<td>( \mathbb{R}^{K \times m} )</td>
<td>Centroid matrix</td>
</tr>
<tr>
<td>( G )</td>
<td>( \mathbb{R}^{K \times K'} )</td>
<td>Alignment matrix</td>
</tr>
<tr>
<td>( W )</td>
<td>( \mathbb{R}^{n \times n} )</td>
<td>Affinity matrix</td>
</tr>
<tr>
<td>( D )</td>
<td>( \mathbb{R}^{n \times n} )</td>
<td>Diagonal summation matrix</td>
</tr>
<tr>
<td>( U )</td>
<td>( \mathbb{R}^{n \times K} )</td>
<td>Scaled indicator matrix</td>
</tr>
</tbody>
</table>

According to the findings on utility function in Chapter 2, we have a new insight of the objective function in Eq. (5.1) as follows.

\[
\min_{H_1, H_2, C, G} \|X_1 - H_1 C\|_F^2 + \|X_2 - H_2 C\|_F^2 + \lambda \|S - H_1 G\|_F^2.
\] (5.3)

5.3 Solutions

In this part, we give the corresponding solution to Eq. (5.2) by derivation, then equivalently transfer the problem into a K-means-like optimization problem in an efficient way.

5.3.1 Algorithm Derivation

To derive the algorithm solving Eq. (5.2), we rewrite Eq. (5.2) as

\[
J = \min_{H_1, H_2, C, G} \text{tr}((X_1 - H_1 C)(X_1 - H_1 C)^\top)
+ (X_2 - H_2 C)(X_2 - H_2 C)^\top + \lambda (S - H_1 G)(S - H_1 G)^\top,
\] (5.4)

where \( \text{tr}(\cdot) \) means the trace of a matrix. By this means, we can update \( H_1, H_2, C \) and \( G \) in an iterative update procedure.
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

**Fixing** $H_1, H_2, G$, **Update** $C$. Let $J_1 = ||X_1 - H_1 C||^2_F + ||X_2 - H_2 C||^2_F$, we have

$$J_1 = \text{tr}((X_1 - H_1 C)(X_1 - H_1 C)^T + (X_2 - H_2 C)(X_2 - H_2 C)^T).$$  \tag{5.5}

Then taking derivative of $C$ and setting it as 0, we get

$$\frac{\partial J_1}{\partial C} = -2H_1^T X_1 + 2H_1^T H_1 C - 2H_2^T X_2 + 2H_2^T H_2 C = 0.$$  \tag{5.6}

Therefore, we can update $C$ as follows:

$$C = (H_1^T H_1 + H_2^T H_2)^{-1}(H_1^T X_1 + H_2^T X_2).$$  \tag{5.7}

**Fixing** $H_1, H_2, C$, **Update** $G$. The term related to $G$ is $||S - H_1 G||^2_F$, then minimize $J_2 = ||S - H_1 G||^2_F$ over $G$, we have

$$J_2 = \text{tr}((S - H_1 G)(S - H_1 G)^T).$$  \tag{5.8}

Next we take the derivative of $J_2$ over $G$, and have

$$\frac{\partial J_2}{\partial G} = -2H_1^T S + 2H_1^T H_1 G = 0.$$  \tag{5.9}

The solution leads to the update rule of $G$ as follows

$$G = (H_1^T H_1)^{-1} H_1^T S.$$  \tag{5.10}

**Fixing** $H_2, G, C$, **Update** $H_1$. The rule of updating $H_1$ is a little different from the above rules, since $H_1$ is not a continues variable. Here we use an exhaustive search for the optimal assignment to find the solution of $H_1$

$$k = \arg \min_j ||X_{1,i} - C_j||^2_F + \lambda ||z_j - H_{1,i} G||^2_F,$$  \tag{5.11}

where $X_{1,i}$ and $H_{1,i}$ denote the $i$-th row in $X_1$ and $H_1$, $C_j$ is the $j$-th centroid and $z_j$ is a $1 \times K$ vector with $j$-th position 1 and others 0.

**Fixing** $H_1, G, C$, **Update** $H_2$. Similar to the update rule of $H_1$, we use the same way to update $H_2$ as follows.

$$k = \arg \min_j ||X_{2,i} - C_j||^2_F.$$  \tag{5.12}

By the above four steps, we alternatively update $C, G, H_1$ and $H_2$ and repeat the process until the objective function converges. Here we decompose the problem into 4 subproblems and each of them is a convex problem with one variable. Therefore, by solving the subproblems alternatively, our method will find a solution with the guarantee of convergence.
5.3.2 K-means-like optimization

Although the above solution is suitable for the clustering with partition level side information, it is not efficient due to some matrix multiplication and inverse. Besides if we have multiple side information, the data are separated to too many fractured pieces, which is hard to operate in real-world applications. This inspires us whether we can solve the above problem in a neat mathematical way with high efficiency. In the following, we equivalently transform the problem into a K-means-like optimization problem via just concatenating the partition level side information with the original data.

First, we introduce the concatenated matrix $D$ as follows,

$$D = \begin{bmatrix} X & S \\ X_2 & 0 \end{bmatrix}.$$

Further we decomposed $D$ into two parts $D = [D_1 D_2]$, where $D_1 = X$ and $D_2 = [S 0]^T$. Here we can see that $D$ is exactly a concatenated matrix with the original data $X$ and partition level side information $S$, $d_i$ consists of two parts, one is the original features $d_i^{(1)} = (d_{i,1}, \cdots, d_{i,m})$, i.e., the first $m$ columns; the other last $K$ columns $d_i^{(2)} = (d_{i,m+1}, \cdots, d_{k,m+K})$ denote the side information; for those instances with side information, we just put the side information behind the original features, and for those instances without side information, zeros are used to filled up.

If we directly apply K-means on the matrix $D$, it might cause some problems. Since we make the partition level side information guide the clustering process in a utility way, those all zeros values should not provide any utility to measure the similarity of two partitions. That is to say, the centroids of K-means is no longer the mean of the data instances belonging to a certain cluster. Let $m_k = (m_k^{(1)}, m_k^{(2)})$ be the $k$-th centroid of K-means, where $m_k^{(1)} = (m_{k,1}, \cdots, m_{k,m})$ and $m_k^{(2)} = (m_{k,m+1}, \cdots, m_{k,m+K})$. We modify the computation of the centroids as follows,

$$m_k^{(1)} = \frac{\sum_{x_i \in C_k} x_i}{|C_k|}, \quad m_k^{(2)} = \frac{\sum_{x_i \in C_k \cap S} x_i}{|C_k \cap S|}. \quad (5.13)$$

Recall that within the standard K-means, the centroids are computed by arithmetic means, whose denominator represents the number of instances in its corresponding cluster. Here in Eq. (5.13), our centroids have two parts $m_k^{(1)}$ and $m_k^{(2)}$. For $m_k^{(1)}$, the denominator is also $|C_k|$; but for $m_k^{(2)}$, the denominator is $|C_k \cap S|$. After modifying the computation of centroids, we have the following theorem.
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

Algorithm 3 The algorithm of PLCC with K-means

**Input:**
- \( X \): data matrix, \( n \times m \);
- \( K \): number of clusters;
- \( S \): \( p \)-partition level side information, \( pn \times K \);
- \( \lambda \): trade-off parameter.

**Output:** optimal \( H^* \);

1. Build the concatenating matrix \( D \), \( n \times (m + K) \);
2. Randomly select \( K \) instances as centroids;
3. repeat
   4. Assign each instance to its closest centroid by the distance function in Eq. (5.15);
   5. Update centroids by Eq. (5.13);
4. until the objective value in Eq. (5.2) remains unchanged.

**Theorem 5.3.1** Given the data matrix \( X \), side information \( S \) and augmented matrix \( D = \{d_i\}_{1 \leq i \leq n} \), we have

\[
\min_{H,C,G} \|X - HC\|_F^2 + \lambda \|S - (H \otimes S)G\|_F^2 \Leftrightarrow \min \sum_{k=1}^{K} \sum_{d_i \in C_k} f(d_i, m_k), \tag{5.14}
\]

where \( m_k \) is the \( k \)-th centroid calculated by Eq. (5.13) and the distance function \( f \) can be computed by

\[
f(d_i, m_k) = \|d_i^{(1)} - m_k^{(1)}\|_2^2 + \lambda I(d_i \in S)\|d_i^{(2)} - m_k^{(2)}\|_2^2. \tag{5.15}
\]

where \( I(d_i \in S) = 1 \) means the side information contains \( x_i \), and 0 otherwise.

**Remark 10** Theorem 5.3.1 exactly maps the problem in Eq. (5.1) into a K-means clustering problem with modified distance function and centroid updating rules, which has a neat mathematical way and can be solved with high efficiency. Taking a close look at the concatenated matrix \( D \), the side information can be regarded as new features with more weights, which is controlled by \( \lambda \). Besides, Theorem 5.3.1 provides a way to clustering with both numeric and categorical features together, which means we calculate the difference between the numeric and categorical part of two instances respectively and add them together.

By Theorem 5.3.1, we transfer the problem into a K-means-like clustering problem. Since the updating rule and distance function have changed, it is necessary to verify the convergency of the K-means-like algorithm.
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

**Theorem 5.3.2** For the objective function in Theorem 5.3.1, the optimization problem is guaranteed to converge in finite two-phase iterations of K-means clustering.

The proof of Theorem 5.2.2 is to show that centroid updating rules in Eq. (5.13) are optimal, which is similar to the proof of Theorem 6 in Ref [51]. We omit the proof here. We summarize the proposed algorithm in Algorithm 3. We can see that the proposed algorithm has the similar structure with the standard K-means, and it also enjoys the almost same time complexity with K-means, $O(tK(n + K))$, where $t$ is the iteration number, $K$ is the cluster number, $n$ and $m$ are the numbers of instance and feature, respectively. Usually $K \ll n$ and $m \ll n$, so the algorithm is roughly linear to the instance number. This indicates that K-means-based PLCC is suitable for large-scale datasets.

5.4 Discussion

In this part, we discuss the extensions of our model. One is to handle multiple partition level side information, the other is to apply spectral clustering with partition level side information.

5.4.1 Handling Multiple Side Information

In real-world application, the side information comes from multiple sources. Thus, how to conduct clustering with multiple side information is common in most scenarios. Next, we modify the objective function to extend our method to handle multiple side information.

\[
\begin{align*}
\min_{H,C,G_j} & \quad \|X - HC\|_F^2 + \sum_{j=1}^{r} \lambda_j \|S_j - (H \otimes S_j)G_j\|_F^2 \\
\text{s.t.} & \quad H_{ik} \in \{0, 1\}, \sum_{k=1}^{K} H_{ik} = 1, 1 \leq i \leq n.
\end{align*}
\]

(5.16)

where $S = \{S_1, S_2, \cdots, S_r\}$ is the set of side information and $\lambda_i$ is the weight of each side information. If we still apply the first solution, the data are separated into so many pieces that it is difficult to handle in practice. Thanks to the K-means-like solution, we concatenate all the side information after the original features and then employ K-means to find the final solution. The centroids consist of $r$ parts, with $m_k = (m_k^{(1)}, m_k^{(2)}, \cdots, m_k^{(r+1)})$, which $m_k^{(j)}, 2 \leq j \leq r + 1$ represents the part of centroids for $r$ side information, and the update rule of centroids and the distance function can be computed as

\[
m_k^{(j+1)} = \frac{\sum_{x_i \in C_k \cap S_j} x_i}{|C_k \cap S_j|},
\]

(5.17)
Algorithm 4 The algorithm of PLCC with spectral clustering

**Input:**
- \( X \): data matrix, \( n \times m \);
- \( K \): number of clusters;
- \( S \): \( p \)-partition level side information, \( pn \times K \);
- \( \lambda \): trade-off parameter.

**Output:** optimal \( H^* \);

1. Build the similarity matrix \( W \);
2. Calculate the largest \( K \) eigenvectors of 
   \[ (D^{-1/2}WD^{-1/2} + \lambda[S0]^{\top}[S0])^{-1/2} \]
3. Run K-means to obtain the final clustering.

\[
f(d_i, m_k) = \|d_i^{(1)} - m_k^{(1)}\|^2 + \sum_{j=1}^{r} \lambda_j 1(d_i \in S_j) \|d_i^{(j+1)} - m_k^{(j+1)}\|^2.
\] (5.18)

### 5.4.2 PLCC with Spectral Clustering

K-means and spectral clustering are two widely used clustering methods, which handle the record data and graph data, respectively. Here we also want to incorporate the partition level side information into spectral clustering for broad use. Here we first give a brief introduction to spectral clustering and extend it to handle partition level side information. Let \( W \) be a symmetric matrix of given data, where \( w_{ij} \) represents a measure of the similarity between \( x_i \) and \( x_j \). The objective function of normalized cuts spectral clustering is the following trace maximization problem \([70]\):

\[
\max_U \text{tr}(U^\top D^{-1/2}WD^{-1/2}U)
\]

s.t. \( U^\top U = I \),

(5.19)

where \( D \) is the diagonal matrix whose diagonal entry is the sum of rows of \( W \) and \( U \) is the scaled cluster membership matrix such that

\[
U_{ij} = \begin{cases} 
1/\sqrt{m_j}, & \text{if } x_i \in C_j \\
0, & \text{otherwise}
\end{cases}
\]

We can easily get \( U = H(H^\top H)^{-1/2} \) and \( U^\top U = I \). The solution is to calculate the largest \( k \) eigenvalues of \( D^{-1/2}WD^{-1/2} \), and run K-means to get the final partition \([70]\).

Similar to the trick we use for K-means, we also separate \( U \) into two parts \( U_1 \) and \( U_2 \) according to side information. Let \( U_1 \) denote the scaled cluster membership matrix for the instances
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

with side information, and $U_2$ represent the scaled cluster membership matrix for the instances without side information. Then we can add the side information part and rewrite Eq. (5.19) as follow.

$$\max_{U_1, U_2} \text{tr}(U_1^\top D^{-1/2} W D^{-1/2} U_1) - \lambda \|S - H_1 G\|^2_F. \quad (5.20)$$

For the second term, through some derivations we can obtain the following equation \[133\],

$$\|S - H_1 G\|^2_F = \|S\|^2_F - \text{tr}(U_1^\top SS^\top U_1). \quad (5.21)$$

Since $\|S\|^2_F$ is a constant, finally we derive the objective function for spectral clustering with partition level side information.

$$\max_{U_1, U_2} \text{tr}(U_1^\top (D^{-1/2} W D^{-1/2} + \lambda \begin{bmatrix} S & S^\top \\ 0 & 0 \end{bmatrix} \begin{bmatrix} S & S^\top \\ 0 & 0 \end{bmatrix}^\top U_1)) \quad (5.22)$$

$$\Leftrightarrow \max_{U} \text{tr}(U^\top (D^{-1/2} W D^{-1/2} + \lambda \begin{bmatrix} S & S^\top \\ 0 & 0 \end{bmatrix}) U).$$

To solve the above optimization problem, we have the following theorem.

**Theorem 5.4.1** The optimal solution $U^*$ is composed by the largest $K$ eigenvectors of $(D^{-1/2} W D^{-1/2} + \lambda \begin{bmatrix} S & S^\top \\ 0 & 0 \end{bmatrix})$.

The proof is similar to the one of spectral clustering, we omit it here due to the limited page. And the algorithm is summarized in Alg. 4.

**Remark 11** Similar to Theorem 5.3.1, Theorem 5.4.1 transforms the spectral clustering with partition level side information into a new spectral clustering problem. So a modified similarity matrix is calculated and followed by the standard spectral clustering. We can see that partition level side information enhances coherence within clusters.
Table 5.2: Experimental Data Sets

<table>
<thead>
<tr>
<th>Data set</th>
<th>#Instances</th>
<th>#Features</th>
<th>#Classes</th>
<th>CV</th>
</tr>
</thead>
<tbody>
<tr>
<td>breast</td>
<td>699</td>
<td>9</td>
<td>2</td>
<td>0.4390</td>
</tr>
<tr>
<td>ecoli*</td>
<td>332</td>
<td>7</td>
<td>6</td>
<td>0.8986</td>
</tr>
<tr>
<td>glass</td>
<td>214</td>
<td>9</td>
<td>6</td>
<td>0.8339</td>
</tr>
<tr>
<td>iris</td>
<td>150</td>
<td>4</td>
<td>3</td>
<td>0.0000</td>
</tr>
<tr>
<td>pendigits</td>
<td>10992</td>
<td>16</td>
<td>10</td>
<td>0.0422</td>
</tr>
<tr>
<td>satimage</td>
<td>4435</td>
<td>36</td>
<td>6</td>
<td>0.4255</td>
</tr>
<tr>
<td>wine+</td>
<td>178</td>
<td>13</td>
<td>3</td>
<td>0.1939</td>
</tr>
<tr>
<td>Dogs</td>
<td>20580</td>
<td>2048</td>
<td>120</td>
<td>0.1354</td>
</tr>
<tr>
<td>AWA</td>
<td>30475</td>
<td>4096</td>
<td>50</td>
<td>1.3499</td>
</tr>
<tr>
<td>Pascal</td>
<td>12695</td>
<td>4096</td>
<td>20</td>
<td>4.6192</td>
</tr>
<tr>
<td>MNIST</td>
<td>70000</td>
<td>160</td>
<td>10</td>
<td>0.0570</td>
</tr>
</tbody>
</table>

*: two clusters containing only two objects are deleted as noise.
+: the last attribute is normalized by a scaling factor 1000.

5.5 Experimental results

In this section, we present the experimental results of PLCC nested K-means and spectral clustering compared to pairwise constrained clustering and ensemble clustering methods. Generally speaking, we first demonstrate the advantages of our method in terms of effectiveness and efficiency. Next, we add noises with different ratios to analyse the robustness and finally the experiments with multiple side information and inconsistent cluster number illustrate the validation of our method in real-world application.

5.5.1 Experimental Setup

*Experimental data.* We use a testbed consisting of seven data sets obtained from UCI repositories\(^1\) and four image data sets with deep features\(^2\)\(^3\)\(^4\)\(^5\). Table 5.2 shows some important characteristics of these datasets, where CV is the Coefficient of Variation statistic that characterizes the degree of class imbalance. A higher CV value indicates a more severe class imbalance.

*Tools.* We choose four methods as competitive methods. LCVEQ\(^115\) is a K-means-based pairwise constraint clustering method; KCC is an ensemble clustering method\(^49\), which first generates one basic partition alone from the data and then fuse this partition with incomplete partition

\(^1\)https://archive.ics.uci.edu/ml/datasets.html
\(^2\)http://vision.stanford.edu/aditya86/ImageNetDogs/
\(^3\)http://attributes.kyb.tuebingen.mpg.de/
\(^4\)https://www.ecse.rpi.edu/homepages/cvrl/database/AttributeDataset.htm
\(^5\)http://yann.lecun.com/exdb/mnist/
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

Table 5.3: Clustering performance on seven real datasets by NMI

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>percent</th>
<th>Ours(SC)</th>
<th>FSC</th>
<th>SC</th>
</tr>
</thead>
<tbody>
<tr>
<td>breast</td>
<td>10%</td>
<td>0.7591</td>
<td>0.5305</td>
<td>0.8071</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>0.7820</td>
<td>0.7430</td>
<td>0.8155</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>0.8071</td>
<td>0.7691</td>
<td>0.8099</td>
</tr>
<tr>
<td></td>
<td>40%</td>
<td>0.8320</td>
<td>0.7973</td>
<td>0.8136</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>0.8538</td>
<td>0.8375</td>
<td>0.8196</td>
</tr>
<tr>
<td>glass</td>
<td>10%</td>
<td>0.6416</td>
<td>0.6184</td>
<td>0.6067</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>0.6820</td>
<td>0.6537</td>
<td>0.6324</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>0.7321</td>
<td>0.6772</td>
<td>0.6782</td>
</tr>
<tr>
<td></td>
<td>40%</td>
<td>0.7692</td>
<td>0.7119</td>
<td>0.7046</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>0.8084</td>
<td>0.7410</td>
<td>0.7283</td>
</tr>
<tr>
<td>iris</td>
<td>10%</td>
<td>0.7653</td>
<td>0.7135</td>
<td>0.7597</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>0.7846</td>
<td>0.7298</td>
<td>0.7829</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>0.8105</td>
<td>0.7846</td>
<td>0.8096</td>
</tr>
<tr>
<td></td>
<td>40%</td>
<td>0.8366</td>
<td>0.7855</td>
<td>0.8033</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>0.8581</td>
<td>0.8067</td>
<td>0.8052</td>
</tr>
<tr>
<td>pendigits</td>
<td>10%</td>
<td>0.6920</td>
<td>0.6808</td>
<td>0.6672</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>0.7101</td>
<td>0.6961</td>
<td>0.6313</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>0.7289</td>
<td>0.7031</td>
<td>0.5984</td>
</tr>
<tr>
<td></td>
<td>40%</td>
<td>0.7456</td>
<td>0.7469</td>
<td>0.5786</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>0.8084</td>
<td>0.7603</td>
<td>0.5406</td>
</tr>
<tr>
<td>satimage</td>
<td>10%</td>
<td>0.6149</td>
<td>0.2318</td>
<td>0.5426</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>0.6143</td>
<td>0.2541</td>
<td>0.5263</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>0.6149</td>
<td>0.3000</td>
<td>0.5133</td>
</tr>
<tr>
<td></td>
<td>40%</td>
<td>0.6153</td>
<td>0.3431</td>
<td>0.4446</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>0.6161</td>
<td>0.4231</td>
<td>0.4505</td>
</tr>
<tr>
<td>wine</td>
<td>10%</td>
<td>0.2944</td>
<td>0.2426</td>
<td>0.2697</td>
</tr>
<tr>
<td></td>
<td>20%</td>
<td>0.3463</td>
<td>0.2321</td>
<td>0.2554</td>
</tr>
<tr>
<td></td>
<td>30%</td>
<td>0.3774</td>
<td>0.2771</td>
<td>0.2339</td>
</tr>
<tr>
<td></td>
<td>40%</td>
<td>0.4350</td>
<td>0.2887</td>
<td>0.1981</td>
</tr>
<tr>
<td></td>
<td>50%</td>
<td>0.4636</td>
<td>0.3267</td>
<td>0.2195</td>
</tr>
</tbody>
</table>

level side information; FSC [129] is a spectral-based clustering method with pairwise constraint; CNMF [121] is an NMF-based constrained clustering method, which also employs the partition level side information as input. In our method, there is only one parameter λ, here we empirically set it to 100, and we also set the weight of side information as 100 in KCC. In the experiments, we randomly select certain percent partition level side information from the ground truth for our method and KCC, then transfer the partition level side information into pairwise constraints for LCVQE and FSC. Although there exist many K-means-based constrained clustering methods, Ref [120] thoroughly studied the K-means-based algorithms for constrained clustering and recommended LCVQE [115], which presents better performance and violates less constraint than CVQE [114] and MPCK-Means [116]. Therefore, we choose LCVQE as the pairwise constraint comparative algorithm. Note that the number of clusters for three algorithms is set to the number of true clusters.

Validation measure. Since class labels are provided for each data set, Normalized Mutual
### Chapter 5. Partition-Level Constraint Clustering

#### Table 5.4: Clustering performance on seven real datasets by $R_n$

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>percent</th>
<th>Ours(K-means)</th>
<th>CNMF</th>
<th>LCVQ</th>
<th>KCC</th>
<th>K-mean</th>
<th>Ours(S)</th>
<th>PSC</th>
<th>SC</th>
</tr>
</thead>
<tbody>
<tr>
<td>breast</td>
<td>10%</td>
<td>0.8564 ± 0.103</td>
<td>0.8271 ± 0.022</td>
<td>0.8562 ± 0.0104</td>
<td>0.8551 ± 0.0090</td>
<td>0.8778 ± 0.0125</td>
<td>0.1112 ± 0.2094</td>
<td>0.8552 ± 0.0000</td>
<td>0.8941 ± 0.0139</td>
</tr>
<tr>
<td>glass</td>
<td>20%</td>
<td>0.9715 ± 0.018</td>
<td>0.9113 ± 0.0145</td>
<td>0.8870 ± 0.0174</td>
<td>0.9174 ± 0.0112</td>
<td>0.9775 ± 0.0137</td>
<td>0.4198 ± 0.0770</td>
<td>0.3570 ± 0.1357</td>
<td>0.3651 ± 0.0808</td>
</tr>
<tr>
<td>ecoli</td>
<td>30%</td>
<td>0.9755 ± 0.032</td>
<td>0.8902 ± 0.0149</td>
<td>0.7743 ± 0.0349</td>
<td>0.6770 ± 0.1666</td>
<td>0.6380 ± 0.1300</td>
<td>0.1437 ± 0.2247</td>
<td>0.5814 ± 0.0984</td>
<td>0.1371 ± 0.0205</td>
</tr>
<tr>
<td>iris</td>
<td>40%</td>
<td>0.9574 ± 0.038</td>
<td>0.5288 ± 0.0317</td>
<td>0.5749 ± 0.0239</td>
<td>0.5204 ± 0.0448</td>
<td>0.3136 ± 0.0627</td>
<td>0.1964 ± 0.1036</td>
<td>0.1902 ± 0.0650</td>
<td>0.1197 ± 0.0661</td>
</tr>
<tr>
<td>pndigits</td>
<td>50%</td>
<td>0.5347 ± 0.006</td>
<td>0.1458 ± 0.0327</td>
<td>0.4660 ± 0.0754</td>
<td>0.4660 ± 0.0494</td>
<td>0.2994 ± 0.0407</td>
<td>0.1807 ± 0.0963</td>
<td>0.2664 ± 0.0258</td>
<td>0.1021 ± 0.0809</td>
</tr>
<tr>
<td>satimage</td>
<td>60%</td>
<td>0.2273 ± 0.043</td>
<td>0.2117 ± 0.0393</td>
<td>0.2029 ± 0.0603</td>
<td>0.1947 ± 0.0463</td>
<td>0.3649 ± 0.1044</td>
<td>0.0717 ± 0.1385</td>
<td>0.3722 ± 0.0669</td>
<td>0.0880 ± 0.1370</td>
</tr>
<tr>
<td>wine</td>
<td>70%</td>
<td>0.2749 ± 0.043</td>
<td>0.1926 ± 0.086</td>
<td>0.1987 ± 0.0697</td>
<td>0.2146 ± 0.0510</td>
<td>0.3068 ± 0.0406</td>
<td>0.2203 ± 0.0555</td>
<td>0.1793 ± 0.0766</td>
<td>0.2465 ± 0.0561</td>
</tr>
</tbody>
</table>

Information (NMI) and Normalized Rand Index $(R_n)$ are used to measure the clustering performance.

**Environment.** All the experiments were run on a Ubuntu 14.04 platform with Intel Core i7-6900K @ 3.2GHz and 64 GB RAM.

#### 5.5.2 Effectiveness and Efficiency

Table 5.3 and 5.4 show the clustering performance of different algorithms on all the seven data sets with side information of different ratios measured by NMI and $R_n$, respectively. In each scenario, 50 runs with different random initializations are conducted and the average performance as well as the standard deviations are reported.

In the K-means-based scenario, our method achieves the best performance in most cases except on glass, pndigits and satimage with 10%, 40% and 50% percent side information (We
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

![Graphs showing impact of \( \lambda \) on NMI and \( R_n \) for satimage and pendigits.](image)

Figure 5.2: Impact of \( \lambda \) on satimage and pendigits.

![Bar charts showing improvement of constrained clustering on glass and wine compared with K-means.](image)

Figure 5.3: Improvement of constrained clustering on glass and wine compared with K-means.

will tune \( \lambda \) to get better performance on pendigits and satimage later). If we take a close look at Table 5.3 and 5.4, our method and KCC keep consistently increasing performance as the percent of side information. LCVQE gets reasonable results on the well separated data sets breast and iris; however, it is surprising that LCVQE gets much worse results with more guidance on glass, pendigits, satimage and wine than the basic K-means without any guidance. This might result from the great
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

Table 5.5: Comparison of Execution Time (in seconds)

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>Ours(K-means)</th>
<th>CNMF</th>
<th>LCVQE</th>
<th>KCC</th>
<th>Ours(SC)</th>
<th>FSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>breast</td>
<td>0.0014</td>
<td>0.4235</td>
<td>0.0461</td>
<td>0.2638</td>
<td>0.5429</td>
<td>4.4632</td>
</tr>
<tr>
<td>ecoli</td>
<td>0.0117</td>
<td>0.1939</td>
<td>0.0318</td>
<td>0.2175</td>
<td>0.1591</td>
<td>1.0187</td>
</tr>
<tr>
<td>glass</td>
<td>0.0052</td>
<td>0.1936</td>
<td>0.0256</td>
<td>0.1263</td>
<td>0.1067</td>
<td>0.3323</td>
</tr>
<tr>
<td>iris</td>
<td>0.0019</td>
<td>0.1259</td>
<td>0.0097</td>
<td>0.0673</td>
<td>0.0874</td>
<td>0.1373</td>
</tr>
<tr>
<td>pendigits</td>
<td>0.4538</td>
<td>195.3840</td>
<td>76.7346</td>
<td>4.9807</td>
<td>651.7113</td>
<td>&gt;4.5hr</td>
</tr>
<tr>
<td>satimage</td>
<td>0.1887</td>
<td>13.8217</td>
<td>11.5499</td>
<td>1.7020</td>
<td>56.7173</td>
<td>1304.2479</td>
</tr>
<tr>
<td>wine</td>
<td>0.0094</td>
<td>0.0535</td>
<td>0.0126</td>
<td>0.1030</td>
<td>0.0718</td>
<td>0.1934</td>
</tr>
</tbody>
</table>

impact of the order of pairwise constraints, which leads to the deformity of clustering structure. In addition, our method enjoys better stability than LCVQE and KCC. For instance, LCVQE has up to 17.5% standard deviation on breast with 50% side information and the volatility of KCC on iris with 20% side information goes up to 16.7%. Fig. [5.3] shows the improvement of constrained clustering algorithms over the baseline methods on glass and wine. It can be seen that for most scenarios, the performance of our method shows a positive relevance with the percentage of side information, which demonstrates the effectiveness of partition level side information. CNMF and our method both take the partition level side information as input. Our method consistently outperforms CNMF, especially on glass and satimages, which demonstrates the utility function helps to preserve the structure from side information. Although we equivalently transfer the partition level side information into pairwise constraints, our clustering method utilizes the consistency within the side information and achieves better results. In the spectral clustering scenario, our method has also consistent better performance than FSC on all datasets but ecoli. Generally speaking, our K-means-based method achieves better performance than the basic K-means, while sometimes our spectral-based method and FSC cannot beat the single spectral clustering.

Next, we evaluate six algorithms in terms of efficiency. Table 5.5 shows the average of execution time of different algorithms with 10% side information. From the table, we can see that our method shows obvious advantages than other three algorithms. On pendigits, our K-means-based method is 10 times faster than KCC, nearly 170 times than LCVQE, 430 times faster than CNMF and our spectral clustering based method run 20 times faster than FSC on large datasets. Taking the effectiveness and efficiency into account, our K-means-based method not only achieves satisfactory result, but also has high efficiency, which verifies that it is suitable for large data set clustering with partition level side information. In the following, we use our K-means-based method as default to further explore its characteristics.

So far, we use a fixed λ to evaluate the clustering performance for fair comparisons due to
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

the unsupervised fashion, and on *pendigits* and *satimage* with 50% side information, our method has a large gap with KCC. In the following, we explore the impact of $\lambda$ on these two data sets. As can be seen in Fig. 5.2 with $\lambda$ varying from $1e+2$ to $1e+6$, KCC keeps stable results with the change of $\lambda$, but suffers from heavy volatility. The performance of our method consistently goes up with the increasing of $\lambda$ with high robustness; besides, our method achieves stability when $\lambda$ is larger than a threshold, like $1e+4$. Recall that $\lambda$ plays a key role in controlling the degree that how the learnt partition achieves close to the side information. From this view, $\lambda$ should be set as large as possible when the given side information is confidence. However, when it comes to noisy side information, we should set $\lambda$ in an appropriate range (See the application in Section 5.6).

5.5.3 Handling Side Information with Noises

In real-world application, the part of side information might be noisy and misleading, thus we validate our method with noisy side information. Here fixing 10% side information, we randomly

Figure 5.4: Impact of noisy side information on *breast* and *pendigits*. 

(a) *breast* by NMI  
(b) *breast* by $R_n$

(c) *pendigits* by NMI  
(d) *pendigits* by $R_n$
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

select certain instances from the side information and randomly label them as noises.

In Fig. 5.4, we can see that the performance of CNMF, LCVQE and KCC drops sharply with the increasing of noise ratio; even 10% noise ratio does great harm to LCVQE on breast. Misleading pairwise constraints and large weight of the noisy side information lead to corrupted results. On the contrary, our method performs high robustness even when the noise ratio is up to 50%. It demonstrates that we do not need exact side information from the specialists, instead a rough good partition level side information is good enough (This point can also be verified in Section 5.6), which validates the effectiveness of our method in practice with noisy side information.

5.5.4 Handling Multiple Side Information

In crowd sourcing, the side information comes from multi-sources and multi-agents. In the following, we show our method handles multiple side information. Here each agent randomly selects 10% instances and provides its corresponding partition level side information. Fig. 5.5 shows the performance of our method with different numbers of side information. With the increasing of the number of side information, the performance on all data sets goes up with a great improvement, even for the not well-separated data sets, such as glass and wine. This reveals that our method can easily be applied to crowd sourcing and significantly improve the clustering result with multiple side information.

5.5.5 Inconsistent Cluster Number

Here we continue to evaluate our proposed method in the scenario that the side information contains inconsistent cluster number with the final cluster number. This obeys the nature of cluster analysis, which aims to uncover the new clusters and cannot be solved by the traditional classification
5.6 Application to Image Cosegmentation

Image clustering, which provides a disjoint image-region partition, has been widely used for the computer vision community, especially the multi-image scenario, such as co-saliency detection [134] and cosegmentation [135, 136, 56]. Here, based on our PLCC method, we propose a Saliency-Guided Constraint Clustering (SG-PLCC) model for the task of image cosegmentation, to show PLCC as an efficient and flexible image clustering tool. In details, we employ saliency prior to obtain the partition level side information, and directly use PLCC to cluster image elements (i.e.,
superpixels) into two classes. In the rest of this section, a brief introduction to the related work comes first, followed by our saliency-guided model, and finally the experimental result is given.

5.6.1 Cosegmentation

Rother et al. [137] first introduced cosegmentation as to extract the similar objects from an image pair with different background, by minimizing the histogram matching in a Markov Random Filed (MRF). The other two early works could be found in [138] and [139], which also focused on the situation of an image pair sharing with the same object. After that, cosegmentation is extended for the multi-image scenario. For example, Joulin et al. [135] employed discriminative clustering to simultaneously segment the foreground from a set of images. For another example, Batra et al. [140] developed an interactive algorithm, intelligently guided by the user scribble information, to achieve cosegmentation for multi-images. Multiple foreground cosegmentation was first proposed by Kim et al. [141] as to jointly segment $K$ different foregrounds from a set of input images. In their work, an iterative optimization process was performed for foreground modeling and region assignment under a greedy manner. Jolin et al. [136] also provided an energy-based model that combines spectral and discriminate clustering to handle multiple foreground and images, and optimized it with Expectation-Minimization (EM) method. Although all these methods above have achieved significant performance, they may suffer from the requirement of user interaction to guide the cosegmentation [140], or the high computing cost of solving an energy optimization [135, 136, 137, 138].

Compared with these works above, the contributions of using PLCC for cosegmentation are threefold: (1) We provide an alternative cosegmentation approach (SG-PLCC), which is simple yet efficient; (2) Our cosegmentation method could be regarded as a rapid preprocessing for other application, benefiting from the linear optimization in PLCC; (3) We provide a flexible framework to integrate various information, such as user scribble, face detection, and saliency prior, which all can be used as the multiple side information for PLCC.

5.6.2 Saliency-Guided Model

Existing saliency models mainly focus on detecting the most attractive object within an image [142], whose output is always a probability distribution map (i.e., saliency map) to the foreground. Thus, it could be seen as a “soft” binary segmentation for an image. Moreover, co-saliency detection [134, 143] aims to extract the common salient objects from multiple images, making it as an appropriate prior for cosegmentation. Generally speaking, there are two main
advantages of using saliency prior: 1) most saliency/co-saliency methods are bottom-up and biology inspired, which means they may detect candidate foreground objects in an unsupervised and rapid way; 2) highlighting the salient objects suppresses the common background across images.

However, there still exist two main problems for directly employing saliency prior as the partition level information. First, saliency detection method only provides the probability of each pixel belonging to the foreground, thus we may need to compute the certain label information based on it. Second, one may note that, the “label” we get from saliency is actually a kind of pseudo label, leading to the fact that our method may suffer from the incorrect label information from the saliency prior.

To solve above challenges, we employ a partial observation strategy. Given \( N \) input images, each of which is represented as a set of superpixels \( X_i = \{x_j\}_{j=1}^n \) by using [144], \( 1 \leq i \leq N \), and assigned a saliency prior by performed any saliency detection algorithm on it. Without loss of generality, we denote \( n \) as the number of superpixels and \( M \) the saliency map for each image. For \( \forall x \in X_i \), let \( M(x) \in [0, 1] \) be its saliency prior, which is computed as the average saliency value of all the pixels within \( x \). Then, the side information \( S \) is defined as:

\[
S(x) = \begin{cases}
2: \text{foreground}, & M(x) \geq T_f \\
1: \text{background}, & M(x) \leq T_b \\
0: \text{missing}, & \text{otherwise}
\end{cases}
\]

(5.23)

where \( T_f \) is a threshold for foreground and \( T_b \) for background. As suggested by [145], \( T_f = \mu + \delta \), where \( \mu \) and \( \delta \) are calculated as the mean and standard deviation of \( M \), respectively. Instead of assigning background to the remainder directly, \( T_b = \mu \) is introduced as a background threshold, that is, we assume the superpixels lower than the average saliency value should belong to the background.
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

Table 5.6: Clustering performance of our method and different priors on iCoseg dataset

<table>
<thead>
<tr>
<th>Criteria</th>
<th>K-means</th>
<th>Saliency Prior</th>
<th>SG-PLCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_n$</td>
<td>0.4311</td>
<td>0.5561 0.5378 0.5215 0.5803</td>
<td>0.6199</td>
</tr>
<tr>
<td>NMI</td>
<td>0.3916</td>
<td>0.4810 0.4762 0.4587 0.5187</td>
<td>0.5534</td>
</tr>
</tbody>
</table>

Table 5.7: Comparison of segmentation accuracy on iCoseg dataset

<table>
<thead>
<tr>
<th>Object class</th>
<th>image subset</th>
<th>[135] [149] [150]</th>
<th>SG-PLCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alaskan Bear</td>
<td>9/19</td>
<td>74.8 90.0 86.4</td>
<td>87.2</td>
</tr>
<tr>
<td>Hot Balloon</td>
<td>8/24</td>
<td>85.2 90.1 89.0</td>
<td>93.8</td>
</tr>
<tr>
<td>Baseball</td>
<td>8/25</td>
<td>73.0 90.1 90.5</td>
<td>92.7</td>
</tr>
<tr>
<td>Bear</td>
<td>5/5</td>
<td>74.0 95.3 80.4</td>
<td>82.3</td>
</tr>
<tr>
<td>Elephant</td>
<td>7/15</td>
<td>70.1 43.1 75.0</td>
<td>90.0</td>
</tr>
<tr>
<td>Ferrari</td>
<td>11/11</td>
<td>85.0 89.9 84.3</td>
<td>90.0</td>
</tr>
<tr>
<td>Gymnastics</td>
<td>6/6</td>
<td>90.9 91.7 87.1</td>
<td>96.9</td>
</tr>
<tr>
<td>Kite</td>
<td>8/18</td>
<td>87.0 90.3 89.8</td>
<td>97.8</td>
</tr>
<tr>
<td>Kite panda</td>
<td>7/7</td>
<td>73.2 90.2 78.3</td>
<td>81.2</td>
</tr>
<tr>
<td>Liverpool</td>
<td>9/33</td>
<td>76.4 87.5 82.6</td>
<td>91.1</td>
</tr>
<tr>
<td>Panda</td>
<td>8/25</td>
<td>84.0 92.7 60.0</td>
<td>80.0</td>
</tr>
<tr>
<td>Skating</td>
<td>7/11</td>
<td>82.1 77.5 76.8</td>
<td>82.2</td>
</tr>
<tr>
<td>Statue</td>
<td>10/41</td>
<td>90.6 93.8 91.6</td>
<td>95.7</td>
</tr>
<tr>
<td>Stone</td>
<td>5/5</td>
<td>56.6 63.3 87.3</td>
<td>82.0</td>
</tr>
<tr>
<td>Stone 2</td>
<td>9/18</td>
<td>86.0 88.8 88.4</td>
<td>80.0</td>
</tr>
<tr>
<td>Taj Mahai</td>
<td>5/5</td>
<td>73.7 91.1 88.7</td>
<td>83.2</td>
</tr>
<tr>
<td>Average</td>
<td>78.9 85.4 83.5</td>
<td>87.9</td>
<td></td>
</tr>
</tbody>
</table>

By using Eq. 5.23, we remain the uncertainty of saliency prior as missing observation, to avoid wrongly labeling the true foreground. On the other hand, some error detections may exist in the saliency prior. We explain these missing labels and possible errors as the noises in side information $S$. As we mentioned in Section 5.5.3 before, PLCC can handle the side information with noises, thus, it alleviates the deficiency of saliency detection.

More details of SG-PLCC are shown by Fig. 5.7. To exploit the corresponding information among input images (a), we perform the co-saliency model proposed by [143] to achieve the saliency prior. After obtaining co-saliency maps (b) and superpixels (c), the side information (d) is computed by Eq. 5.23. We then simply extract the mean Lab feature for each superpixel in (e). Finally, the cosegmentation (f) is achieved by performing PLCC for each image, which jointly combines the feature and label information. It worthy to note that, most missing observations in (d) are segmented as foreground successfully, showing the capability of PLCC to handle the noise in side information.
5.6.3 Experimental Result

Here, we test the effectiveness of the proposed clustering approach PLCC for a real application task (i.e., image cosegmentation). We perform our cosegmentation model SG-PLCC on the widely used iCoseg dataset [140], which consists of 643 images with 38 object groups and focuses on the foreground/background segmentation.

Implementation Details. The saliency prior is obtained by conducting the co-saliency model in [143], which combines the results of three efficient saliency detection methods [146, 147, 148]. For simplicity, our SG-PLCC approach employs the LAB features on a superpixel level, i.e., the mean LAB color values (three-dimensional vector) of a superpixel. Three baseline methods [135, 149, 150] are used to compare with our SG-PLCC, where we directly report the results provided in their papers.

Clustering Performance. As shown by Table 5.6, we fist validate our result as a $K = 2$ clustering task, under two criteria $R_n$ and NMI, respectively. A classic $K$-means algorithm is directly employed with Lab color feature on image superpixels as a baseline. However, it cannot explore the clustering structure effectively. On the other side, we divide each saliency map [143] (including three elementary methods [146, 147, 148]) into 2 classes with $T_f$ thresholding, to demonstrate the effectiveness of our saliency prior. Interestingly, though the discriminative of feature is limited, our SG-PLCC model still improves the performance of saliency prior $S$ by around 4%, showing that the PLCC can combine the feature and side information effectively.

Cosegmentation Performance. Table 5.7 shows the quantitative comparison between SG-PLCC and other methods by segmentation accuracy (i.e., the percentage of correctly classified pixels to the total). We follow the same experiment setting as [149], where all the methods are tested on a subset of each image group from 16 selected object classes in the iCoseg dataset. For fairness, we average the performance of SG-PLCC over 20 random image subset for each object. It can be seen that, SG-PLCC outperforms others in general, and improves the average accuracy of 2.5% to the second best. Moreover, our method achieves 95.7%, 96.9% and 97.8%, nearly one hundred percentage, on the classes of Statue, Gymnastics, and Kite, respectively, without high computing optimization and label information, which significantly shows the success of using PLCC for real application.

Visually, some examples of our results are shown in Fig. 5.8, where the foreground is segmented with yellow line while the background darkened for a better view. For these cases, pretty fine segmentations are provided by SG-PLCC. However, our performance may degrade for some more challenging scenarios. As shown by Fig. 5.9, we fail to segment out the entire foreground, and
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

Figure 5.8: Cosegmentation results of SG-PLCC on six image groups.

suffer from the cluttered background. To solve these problems, we could feed the SG-PLCC results into some conventional segmentation frameworks to improve the performance of cosegmentation, and employ more discriminative feature rather than the raw Lab. In addition, the SG-PLCC model can be easily extended for the multi-class cosegmentation with the increase of clustering number.

To sum up, the proposed SG-PLCC model provides an example of using PLCC in the real application task. Although SG-PLCC is directly performed with raw features, and only guided by unsupervised saliency prior, we still achieve a promising result for image cosegmentation, which demonstrates the power of our PLCC method.
CHAPTER 5. PARTITION-LEVEL CONSTRAINT CLUSTERING

Figure 5.9: Some challenging examples for our SG-PLCC model.

5.7 Summary

In this chapter, we proposed a novel framework for clustering with partition level side information, called PLCC. Different from pairwise constraints, partition level side information accords with the labeling from human being with other instances as references. Within the PLCC framework, we formulated the problem via conducting clustering and making the structure agree as much as possible with side information. Then we gave its corresponding solution, equivalently transformed it into K-means clustering and extended it to handle multiple side information and spectral clustering. Extensive experiments demonstrated the effectiveness and efficiency of our method compared to three state-of-the-art algorithms. Besides, our method had high robustness when it comes to noisy side information and finally we validated the performance of our method with multiple side information and inconsistent cluster number setting. The cosegmentation application demonstrated the effectiveness of PLCC as a flexible framework in the image domain.
Chapter 6

Structure-Preserved Domain Adaptation

Domain adaptation, as a branch of transfer learning, has attracted lots of attention recently [151] and has been widely discussed in data mining tasks [152]. Basically, it manages to adapt feature spaces of different domains, but of the same or similar tasks. A good instance would be adapting the object classifier trained from low-resolution webcam images for the image recognition of the same category captured by high-resolution digital cameras. The challenge lies in the significantly different appearances between webcam and digital camera images due to image resolutions.

In domain adaptation, we denote domains with well-labeled data as source domains while the domain being classified as the target domain. Most domain adaptation algorithms manage to align them so that the well-established knowledge can be transferred from source to target domain. Briefly, these algorithms are characterized by the following two groups: (1) feature space adaptation, (2) classifier adaptation. Research work regarding to feature space adaptation seeks for a common subspace where the feature space divergence between source and target domains is minimized [153, 154, 155, 156, 157, 158, 159, 160]. However, as fewer target labels are available in the training, they may not be able to employ discriminant knowledge, and thus fail to achieve conditional distribution alignment. This becomes extremely challenging for multiple source data. On the other hand, classifier adaptation usually adapts the classifier learned in the source, e.g., SVM, to the target data [161, 162, 163]. Usually a classifier is learnt in the common space with source data, then predicts the labels for the data from the target domain. Apparently, such techniques require target labels for classifier adaptation, and therefore are inappropriate for unsupervised domain adaptation. While considerable endeavor has been made to domain adaptation, it concentrates more on the single source domain adaptation [159, 164, 156, 159]. Even worse, for classifier adaptation, only the knowledge derived from the hyperplane is transferred to the target domain and the global structure
information of the source domain is ignored. In fact, the performance of existing multi-source domain adaptation methods is far from satisfactory and is even worse than those single source domain adaptation methods. Thus, multi-source domain adaptation remains a critical challenge in community of transfer learning.

In this chapter, we target at the challenging unsupervised domain adaptation problem, given the unavailable target labels and complex composition of single or multiple source domains. To that end, we propose a novel semi-supervised clustering framework to both preserve the intrinsic structures of source and target domains and predict the labels of target domain. We employ semi-supervised clustering in two source domains together with the target domain, while ensuring the label consistency at the partition level for the unknown target data. Specifically, the source and target data are put together for clustering, which explores the structures of source and target domains and keeps the structures of source domains consistent with the label information as much as possible. The consistent label information from source domains can further guide the process of the target domain clustering. In this way, we cast the original single or multiple source domain adaptation to a joint semi-supervised clustering with common unknown target labels and known multiple source labels. To the best of our knowledge, this is the first work to formulate unsupervised domain adaptation into a semi-supervised clustering framework. Then we derive the algorithm by taking the derivatives and give its corresponding solution. Furthermore, a K-means-like optimization solution is further designed to the proposed method in a neat mathematical and highly efficient way. Extensive experiments on two popular domain adaptation databases demonstrate the effectiveness of our method against the most recent state-of-the-art methods by a large margin. Our method yields competitive performance in the single source setting compared with the state-of-the-art, and excels others by a large margin in the multi-source setting, which verifies that the structure-preserved clustering can take use of multi-source domains and achieve robust and high performance compared with single source domain adaptation. We highlight our main contributions as follows.

- We propose a novel constrained clustering algorithm for single or multiple source domain adaptation. Specially we put the source and target data together for clustering. Not only the structure of target domain is explored, but also the structures of source domains are consistently kept with the label information as much as possible, which can further guide the target domain clustering.

- By introducing an augmented matrix, a K-means-like optimization is nontrivially designed with modified distance function and update rule for centroids in an efficient way.
CHAPTER 6. STRUCTURE-PRESERVED DOMAIN ADAPTATION

- Extensive experiments on two popular domain adaptation databases demonstrate the effectiveness of our method against the most recent state-of-the-art methods by a large margin, which verifies the effectiveness of structure-preserved clustering for unsupervised domain adaptation.

6.1 Unsupervised Domain Adaptation

Unsupervised domain adaptation has been raised recently for the scenario where no target labels are available given the shared class information across domains [157, 158]. When doing classification, source data are directly adopted as references. Feature adaption is one of the typical methods to address the domain shift, include searching intermediate subspaces that describe the smooth transition from one domain to another [157, 158, 166] and learning common feature space [153, 156, 155, 160, 167]. Among them, LSTL [160] and JDA [159] are two typical subspace based domain adaption algorithms. Hou et al. even proposed by involving the pseudo target labels optimization to further consider the conditional distribution alignment under the common subspace [168]. On the other hand, classifiers can also be adapted based on the source and target data, however, existing works in this line are not appropriate for unsupervised domain adaptation without target labels [161, 162, 163].

In domain adaptation, multi-source domains are even more critical than single source domain, as it needs to handle both domain alignment between source and target and that within different source domains [169]. Existing works being able to tackle multi-source domain usually naively mix all source data and treat all of them equally [157, 158, 170]. Therefore, they are not able to explore the underlying structure of each domain, and introduce negative transfer due to the complex composition of multiple domains. Recently, researchers propose a few methods to reshape the multiple sources by discovering latent domains. For example, they re-organized data according to the modality information and formulate a new constrained clustering method [171] to discover latent domains. Another example is to mine the latent domains under certain criteria such as domain integrity and separability [172]. RDALR [173] and SDDL [170] are two typical multi-source domain adaptation models, which aim to transform sources into a new space with a reconstruction formulation in a low-rank or sparse constraint. Although there are studies on multi-source domain adaptation [174], most of them require target labels for classifier adaptation, which is different from unsupervised domain adaptation problem setting.

Most recently, deep transfer learning algorithms are developed to generalize deep structure to the transfer learning scenario [175, 176, 177, 167, 178]. The main idea is to enhance the feature
CHAPTER 6. STRUCTURE-PRESERVED DOMAIN ADAPTATION

transferability in the task-specific layers of the deep neural networks by explicitly reducing the domain discrepancy. In this way, the obtained feed-forward networks can be applicable to the target domain without being hindered by the domain shift. For example, Long et al. explored multi-layer adaptation on the fully-connected layers for source and target networks, and therefore, the new designed loss would help solve the domain mismatch during network learning [176]. However, those algorithms all manage to reduce the marginal distribution divergence across two domains, which fails to uncover the intrinsic class-wise structure of two domains.

6.2 SP-UDA Framework

Typically, domain adaptation aims to borrow some well-defined knowledge in the source domain and apply it to the task on the target domain [164]. Here source domain and target domain are different but related. The goal of domain adaptation is to make use of the data and labels in the source domains to predict the labels for the target domain.

Since the distributions of data from source and target domains have large divergences, the alignment of two distributions is regarded as the key problem in domain adaptation area. In light of this, tremendous efforts have been taken to seek a common space. After that, a classifier learnt with the source data and the corresponding labels can be adapted to the task on target data. Admittedly, the alignment is crucial to the success of domain adaptation. However, how to effectively transfer the knowledge from source domain to the target domain is another key factor, which is unfortunately usually being ignored.

Most of existing works train a classifier in the common space with the source data and apply it for target domain. In such a way, only several points in the source domain play the determined role for the hyper-plain of the classifier and other points are not utilized effectively. To cope with this challenge, we focus on the way of knowledge transfer for domain adaptation. Specifically, a partition-level constraint is employed to preserve and transfer the whole source structure and then the source and target data are put together as a constrained clustering problem.

Without loss of generality, suppose we have the source data with label information and target data without label information, and our task is to assign labels for the target data. Let $X_S$ denote the data matrix of the source domain with $n_S$ and $m$ features, $Y_S$ is 1-of-$K$ coding label matrix of source data, where $K$ is the number of classes; $X_T$ represents the data matrix of target domain with $n_T$ instances and $m$ features. Since our goal is to effectively transfer the knowledge from source domain to the target domain, rather than the alignment the distributions of different domains, here we
CHAPTER 6. STRUCTURE-PRESERVED DOMAIN ADAPTATION

assume that the alignment projection $P$ is pre-known or pre-learned that $Z_S = X_S P$, $Z_T = X_T P$.

In the following, we introduce our problem definition and give the framework of Structure-Preserved Unsupervised Domain Adaptation (SP-UDA).

6.2.1 Problem Definition

The alignment and transfer are two key challenges in domain adaptation, and we focus on the second one. Previous works make use of the hyperplane learnt from source domain to predict the labels for target domain. Only some key instances determine the hyperplane, while other instances are not fully utilized. Here our goal of domain adaptation is to predict the labels of the target domain as well as to keep the intrinsic structures of both source and target domains. That means the whole structure of source domain is taken full use for this task.

Moreover, although many efforts have been taken in this field and some reasonable performance has been achieved, most existing work pays more attention to the single source domain adaptation \[159, 164, 156\]. For the methods, which can handle multi-source domain adaptation, the performance is far from satisfactory (See Table \[6.2\]), or even worse than the single source domain adaptation. In light of this, we also take the multi-source domain adaptation into consideration in a unified framework. Therefore, we formalize the problem definition as follows:

- How to incorporate the structure of source domain to predict the labels of target domain?
- How to conduct multi-source domain adaptation in a unified framework?
- How to provide a neat formulation and its corresponding solution?

6.2.2 Framework

In order to capture the structure of different domains, we formulate the problem as a clustering problem. Generally speaking, the source and target data are put together for clustering, which explores the structures of target domain as well as keeps the structures of source domains consistent with the label information as much as possible. Here we propose the framework of Structure-Preserved Unsupervised Domain Adaptation (SP-UDA). Table \[6.1\] provides the key variables used along this paper. Given the pre-learnt alignment projection $P$, we have the new representation in the common space of source and target data as $Z_S$ and $Z_T$. Our goal is to utilize the whole structure of source domain for the recognition of target data. To achieve this, the source and target data are put together for clustering with the partition-level constraint from the source data label, which
preserves the whole source structure and further guides the clustering process. The SP-UDA can be summarized as follows:

$$\min_{H_S, H_T} \mathcal{J}(Z_S, Z_T; K) - \lambda U_c(H_S, Y_S),$$  \hspace{1cm} (6.1)$$

where $\mathcal{J}$ is the objective function of certain clustering algorithm, which takes $Z_S$ and $Z_T$ as the input, partitions the data into $K$ clusters and returns the assignment matrices $H_S$ and $H_T$; $U_c$ is the well-known categorical utility function $^{[29]}$, which treats the similarity of two partitions.

The benefits of the SP-UDA framework in Eq. (6.1) lie in that (1) we employ the constrained clustering approach instead of classification for the recognition of target data, so that these target data without labels are involved during the training process, (2) the categorical utility function plays as the partition-level constraint, which not only preserves and transfers the whole source structure to target data, but also guides the target data clustering and (3) the framework can be efficiently solved via a K-means-like solution, if we choose K-means as the core clustering algorithm in $\mathcal{J}$, which will be further discussed in Section 6.2.5.

Note that in our SP-UDA framework, we assume that the projection $P$ from the original feature space to the common space is known, and the inputs are $Z_S = X_S P$, $Z_T = X_T P$, the source and target data matrix after the projection $P$. Actually, there are tremendous efforts to address the projection problem, such as Geodesic Flow Kernel (GFK) $^{[158]}$, Transfer Component Analysis (TCA) $^{[164]}$, Transfer Subspace Learning (TSL) $^{[156]}$ and Joint Domain Adaptation (JDA) $^{[159]}$, where the projection $P$ learnt from these algorithms plays a role in aligning the data from source and target domain into a common space and it preserves the cluster structure to some extent. Although we can involve the projection learning within our SP-UDA framework, combining some mature techniques is not our selling point and this also leads our model complex and loses the neat formulation. In this paper, we focus on the structure-preserved learning to enhance the

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_S$</td>
<td>Source domain data matrix in the original feature space</td>
</tr>
<tr>
<td>$Y_S$</td>
<td>Source domain indicator matrix</td>
</tr>
<tr>
<td>$X_T$</td>
<td>Target domain data matrix in the original feature space</td>
</tr>
<tr>
<td>$K$</td>
<td>Number of clusters</td>
</tr>
<tr>
<td>$P$</td>
<td>Projection from original space to common space</td>
</tr>
<tr>
<td>$Z_S$</td>
<td>Source domain data matrix in the aligned feature space</td>
</tr>
<tr>
<td>$Z_T$</td>
<td>Target domain data matrix in the aligned feature space</td>
</tr>
<tr>
<td>$H_S$</td>
<td>Learnt source domain indicator matrix</td>
</tr>
<tr>
<td>$H_T$</td>
<td>Learnt target domain indicator matrix</td>
</tr>
</tbody>
</table>
CHAPTER 6. STRUCTURE-PRESERVED DOMAIN ADAPTATION

domain adaptation performance. Therefore, we directly start from source and target data matrix after the projection. In the following, we introduce how to apply the SP-UDA framework for single and multi-source domain adaptation.

6.2.3 SP-UDA for Single Source Domain

Here we illustrate how to apply the SP-UDA framework for single source domain adaptation. For similarity, we choose K-means as the core clustering algorithm in $J$, which leads the following objective function:

$$
\min \left \| \begin{bmatrix} Z_S \\ Z_T \end{bmatrix} - \begin{bmatrix} H_S \\ H_T \end{bmatrix} G \right \|_F^2 - \lambda U_c(H_S, Y_S),
$$

where $Z_S, Z_T, Y_S$ are input variables, $H_S$ and $H_T$ are the unknown assignment matrices for source and target data, respectively, and $G$ is the corresponding centroids matrix.

The above objective function consists of two parts. One is the standard K-means with squared Euclidean distance for the combined source and target data, the other is a term measuring the disagreement between the indicator matrix $H_S$ and the label information of the source domains. After the projection, the source and target data $Z_S$ and $Z_t$ are aligned in the common space. Data points with the same label, no matter from the source domain or target domain form a cluster and they have the same cluster centroid. Therefore, we employ $K$ centroids $G$ to represent all the data points in the aligned space, where $H_S$ and $H_T$ are the indicator matrices to indicate the data point belonging to the nearest centroid in $G$. The two terms in Eq. (6.2) share different functions. The K-means term aims to explore the combined source and target data structure, while the categorical utility function is expected to make the learnt source structure be similar to the source labels as much as possible in order to preserve the source structure, where it plays a role in uncovering the target structure with the guidance of source structure.

Here we aim to find a solution containing $H_S$ and $H_T$, which not only explores the intrinsic structural information from target data, but also keeps the structures of several source domains. Unlike existing works where only the knowledge from the hyperplane is transferred to predict the labels for target domain, here we transfer the whole structures from several different domains to enhance the task on target domain.

According to the findings in Chapter 2, we have a new formulation of the problem in
CHAPTER 6. STRUCTURE-PRESERVED DOMAIN ADAPTATION

Eq. (6.2) as follows:

\[
\min \left\| \begin{bmatrix} Z_S \\ Z_T \end{bmatrix} - \begin{bmatrix} H_S \\ H_T \end{bmatrix} G \right\|_F^2 + \lambda \left\| Y_S - H_S M \right\|_F^2.
\]  

(6.3)

In Eq. (6.3), \( M \) plays a role in shuffling the order of clusters in \( Y_S \). It is crucial to align two partitions due to the non-ordering of cluster labels. For instance, the distance between two exact same partitions with different label orders cannot be zero without alignment. Although one variable \( M \) is involved in Eq. (6.3), we can seek the solution by iteratively updating each unknown continuous variable by taking derivation and greedy search for the discrete variables.

**Fixing others, Update** \( G \). Let \( Z = [Z_S; Z_T] \) and \( H = [H_S; H_T] \), then the term related to \( G \) is \( J_1 = \|Z - H G\|_F^2 \). By taking the derivative of \( J \) over \( G \), we have

\[
\frac{\partial J_1}{\partial G} = -2H^T Z + 2H^T H G = 0.
\] 

(6.4)

The solution leads to the update rule of \( G_1 \) as follows.

\[
G = (H^T H)^{-1} H^T Z.
\] 

(6.5)

**Fixing others, Update** \( M \). Let \( J_2 = \|Y_S - H_S M\|_F^2 \) and minimize \( J_2 \) over \( M \) by taking the derivative, we have

\[
\frac{\partial J_2}{\partial M} = -2H_S^T Y_S + 2H_S^T H_S M = 0.
\] 

(6.6)

Thus, we have the following update rule for \( M \) as:

\[
M = (H_S^T H_S)^{-1} H_S^T Y_S.
\] 

(6.7)

**Fixing others, Update** \( H_S \). The rules of updating \( H_S \) is slightly different from the above rules. Due to the discrete variable, here we use an exhaustive search for the optimal assignment to find the solutions for each data point in \( H_S \) as follows:

\[
k = \arg \min_j \|Z_{S,i} - G_j\|_2^2 + \lambda \|Y_{S,i} - b_j M\|_2^2,
\] 

(6.8)

where \( Z_{S,i} \) and \( Y_{S,i} \) denote the \( i \)-th row in \( Z_S \) and \( Y_S \), \( G_j \) is the \( j \)-th centroid or row of \( G \) and \( b_j \) is a \( 1 \times K \) vector with \( j \)-th position 1 and others 0.

**Fixing others, Update** \( H_T \). For \( H_T \), similarly we apply an exhaustive search for each data point in \( H_T \),

\[
k = \arg \min_j \|Z_{T,i} - G_j\|_2^2,
\] 

(6.9)
Algorithm 5 The algorithm of SP-UDA for single source domain.

**Input:** \( Z_S, Z_T \): data matrix;
\( Y_S \): the labels of source domains;
\( K \): number of clusters;
\( \lambda \): trade-off parameter.

**Output:** optimal \( H_S, H_T \);

1. Initialize \( H_S \) and \( H_T \);
2. repeat
3. Update \( G \) by Eq. (6.5);
4. Update \( M \) by Eq. (6.7);
5. Update \( H_S \) and \( H_T \) by Eq. (6.8) and (6.9), respectively;
6. until the objective value in Eq. (6.2) remains unchanged.

where \( Z_{T,i} \) denotes the \( i \)-th row in \( Z_T \) and \( G_j \) is the \( j \)-th centroid or row in \( G \).

The algorithm by derivation is given in Algorithm 5. We decompose the problem into several sub-problems, which have the closed-form solutions. Therefore, the final solution can be guaranteed to converge to the local minimum. In essence, Algorithm 5 is a constrained clustering method. Different from the traditional constrained clustering algorithms, which employs the pairwise cannot-link or must-link constraints to shape the cluster structure, here a novel partition-level constraint \([118]\) is applied here to treat the source structure as a whole and preserve the whole structure during the clustering process. This further guides the target data clustering. Although the update rule in Eq. (6.9) seems not to include \( Y_S \), the source structure affects the assignment matrix \( H_S \) and further conducts on the centroid matrix \( G \) in the common space. This indicates that \( Y_S \) helps to seek the better cluster centers in the common space, which facilitates the target data clustering.

### 6.2.4 SP-UDA for Multiple Source Domains

Next we continue to apply the SP-UDA framework for single source domain adaptation. Without loss of generality, suppose we have the two source domains and one target domain. With some alignment projections \( P_1 \) and \( P_2 \), we have the common features \( Z_{S_1} = X_{S_1} P_1, Z_{T_1} = X_T P_1, Z_{S_2} = X_{S_2} P_2 \) and \( Z_{T_2} = X_T P_2 \). Our goal is to fuse the information from multi-source domain to provide better performance on target domain. Here suppose that the alignment projects \( P_1 \) and \( P_2 \) are given, we start from the \( Z_{S_1}, Z_{S_2}, Z_{T_1} \) and \( Z_{T_2} \) to predict the labels \( H_T \) for target domain. In the
following, we first give the objective function for two source domains in the SP-UDA and provide the corresponding solution.

Here we directly give the following objective function for two source domains scenario.

\[
\begin{align*}
\min \| & \left[ \begin{array}{c} Z_{S_1} \\ Z_{T_1} \end{array} \right] - \left[ \begin{array}{c} H_{S_1} \\ H_T \end{array} \right] G_1 \|_F^2 + \lambda \| Y_{S_1} - H_{S_1} M_1 \|_F^2 \\
+ & \| \left[ \begin{array}{c} Z_{S_2} \\ Z_{T_2} \end{array} \right] - \left[ \begin{array}{c} H_{S_2} \\ H_T \end{array} \right] G_2 \|_F^2 + \lambda \| Y_{S_2} - H_{S_2} M_2 \|_F^2,
\end{align*}
\]

(6.10)

where \(Z_{S_1}, Z_{S_2}, Z_{T_1}, Z_{T_2}, Y_{S_1}\) and \(Y_{S_2}\) are input variables, the rest are unknown. \(H_{S_1}, H_{S_2}\) and \(H_T\) are the indicator matrices for two source domains and the target domain respectively, \(G_1\) and \(G_2\) are the corresponding centroids matrices, \(M_1\) and \(M_2\) are two alignment matrices to match \(Y_{S_1}\) and \(Y_{S_2}\), respectively.

Since the problem in Eq. (6.10) is not jointly convex to all the variables, here we iteratively update each unknown variable by taking derivation.

**Fixing others, Update** \(G_1, G_2\). Let \(Z_1 = [Z_{S_1}; Z_{T_1}]\) and \(H_1 = [H_{S_1}; H_T]\), then the term related to \(G_1\) is \(J_1 = \| Z_1 - H_1 G_1 \|_F^2\). By taking the derivative of \(J_1\) over \(G_1\), we have

\[
\frac{\partial J_1}{\partial G_1} = -2H_1^\top Z_1 + 2H_1^\top H_1 G_1 = 0.
\]

(6.11)

The solution leads to the update rule of \(G_1\) as follows.

\[
G_1 = (H_1^\top H_1)^{-1} H_1^\top Z_1.
\]

(6.12)

Similarly, \(Z_2 = [Z_{S_2}; Z_{T_2}]\) and \(H_2 = [H_{S_2}; H_T]\), we have the following rule to update \(G_2\).

\[
G_2 = (H_2^\top H_2)^{-1} H_2^\top Z_2.
\]

(6.13)

**Fixing others, Update** \(M_1, M_2\). Let \(J_2 = \| Y_{S_1} - H_{S_1} M_1 \|_F^2\) and minimize \(J_2\) over \(M_1\) by taking the derivative, we have

\[
\frac{\partial J_2}{\partial M_1} = -2H_{S_1}^\top Y_{S_1} + 2H_{S_1}^\top H_{S_1} M_1 = 0.
\]

(6.14)

The update rule of \(M_2\) is similar to the one of \(M_1\), so we have the following update rules.

\[
M_1 = (H_{S_1}^\top H_{S_1})^{-1} H_{S_1}^\top Y_{S_1},
\]

\[
M_2 = (H_{S_2}^\top H_{S_2})^{-1} H_{S_2}^\top Y_{S_2}.
\]

(6.15)
Algorithm 6 The algorithm of SP-UDA for multiple source domains.

Input: \( Z_{S_1}, Z_{T_1}, Z_{S_2}, Z_{T_2} \): data matrix;
\( Y_{S_1}, Y_{S_2} \): the labels of source domains;
\( K \): number of clusters;
\( \lambda \): trade-off parameter.

Output: optimal \( H_{S_1}, H_{S_2}, H_T \);

1. Initialize \( H_{S_1}, H_{S_2} \) and \( H_T \);
2. repeat
3. Update \( G_1 \) and \( G_2 \) by Eq. (6.12) and (6.13);
4. Update \( M_1 \) and \( M_2 \) by Eq. (6.15);
5. Update \( H_{S_1}, H_{S_2} \) and \( H_T \) by Eq. (6.16), (6.17) and (6.18), respectively;
6. until the objective value in Eq. (6.10) remains unchanged.

Fixing others, Update \( H_{S_1}, H_{S_2} \). The rules of updating \( H_{S_1} \) and \( H_{S_2} \) are slightly different from the above rules, since they are not continuous variables. Here we use a exhaustive search for the optimal assignment to find the solutions.

For \( H_{S_1} \), we have
\[
k = \arg \min_j ||Z_{S_1,i} - G_{1,j}||_2^2 + \lambda ||Y_{S_1,i} - b_j M_1||_2^2, \tag{6.16}
\]
where \( Z_{S_1,i} \) and \( Y_{S_1,i} \) denote the \( i \)-th row in \( Z_{S_1} \) and \( H_{S_1} \), \( G_{1,j} \) is the \( j \)-th centroid of \( G_1 \) and \( b_j \) is a \( 1 \times K \) vector with \( j \)-th position 1 and others 0.

For \( H_{S_2} \), we have
\[
k = \arg \min_j ||Z_{S_2,i} - G_{2,j}||_2^2 + \lambda ||Y_{S_2,i} - b_j M_2||_2^2, \tag{6.17}
\]
where \( Z_{S_2,i} \) and \( Y_{S_2,i} \) denote the \( i \)-th row in \( Z_{S_2} \) and \( H_{S_2} \), \( G_{2,j} \) is the \( j \)-th centroid of \( G_2 \) and \( b_j \) is a \( 1 \times K \) vector with \( j \)-th position 1 and others 0.

Fixing others, Update \( H_T \). For \( H_T \), we still use an exhaustive search for the solution,
\[
k = \arg \min_j ||Z_{T_1,i} - G_{1,j}||_2^2 + ||Z_{T_2,i} - G_{2,j}||_2^2, \tag{6.18}
\]
where \( Z_{T_1,i} \) and \( Z_{T_1,j} \) denote the \( i \)-th row in \( Z_{T_1} \) and \( Z_{T_2} \), and \( G_{1,j}, G_{2,j} \) are the \( j \)-th centroid of \( G_1, G_2 \).

The algorithm by derivation is given in Algorithm 6. We decompose the problem into several sub-problems, which has the closed-form solutions. Therefore, the final solution can be
guaranteed to converge to the local minimum. Although we can take the derivative of each unknown variable to obtain the solution, it is not efficient due to the matrix product and inverse. Besides if we have several source domains, a lot of variables need to be updated, which is hard to operate in real-world applications. This motivates us to solve the above problem in a neat mathematical way with high efficiency. In the following, we equivalently transfer the problem into a K-means-like optimization problem via an augmented matrix.

6.2.5 K-means-like Optimization

In the above two sections, we apply the derivatives and greedy search for the solution. However, we find that when the number of source domains increases, the solution requests many variables to be updated, which makes the model fragmented and inefficient. To cope with this challenge, we equivalently transfer the problem into a K-means like optimization problem in a neat and efficient way. Generally speaking, a K-means-like solution is designed with neat mathematical formulation by introducing an augmented matrix and the convergence of the new solution is guaranteed. The discussion on the time complexity is also provided for fully understanding the solution.

Before giving the K-means-like optimization, we first introduce the augmented matrix $D$ as follows:

$$
D = \begin{pmatrix}
Z_{S_1} & Y_{S_1} & 0 & 0 \\
0 & 0 & Z_{S_2} & Y_{S_2} \\
Z_{T_1} & 0 & Z_{T_2} & 0
\end{pmatrix},
$$

(6.19)

where $d_i$ is the $i$-th row of $D$, which consists of four parts. The first one is the features $d_i^{(1)} = (d_{i,1}, \cdots, d_{i,m})$ after projection $P_1$, the next $K$ columns $d_i^{(2)} = (d_{i,m+1}, \cdots, d_{i,m+K})$ denotes the label information of the first source domain, while the third and fourth parts denote the features and labels of the second domain. From Eq. (6.19), we can see that each row denotes each domain and the first and third columns represent the common spaces between two source domains and target domain, respectively, while the second and fourth columns represent the label information of each domain. Zeros are used to fill up the other parts of the augmented matrix.

By these means, we formulate the problem as a semi-supervised clustering with missing values. If we just apply K-means on the matrix $D$, there will be some problems. Zeros are the artificial features, rather than the true values so that all zero values contribute to the computation of the centroids, which inevitably interferes the final cluster structure. Since we make the label information from two source domains guide the clustering process in a utility way, those all zeros values will not
Algorithm 7 The algorithm of SP-UDA for multiple source domains via K-means-like optimization

**Input:** $Z_{S_1}, Z_{T_1}, Z_{S_2}, Z_{T_2}$: data matrix;
$Y_{S_1}, Y_{S_2}$: the labels of source domains;
$K$: number of clusters;
$\lambda$: trade-off parameter.

**Output:** optimal $H_{S_1}, H_{S_2}, H_T$;

1. Build the concatenating matrix $D$;
2. Randomly select $K$ instances as centroids;
3. repeat
   4. Assign each instance to its closest centroid by the distance function in Eq. (6.22);
   5. Update centroids by Eq. (6.20);
4. until the objective value in Eq. (6.10) remains unchanged.

provide any utility to measure the similarity of two partitions. That is to say, the centroids of K-means is no longer the mean of the data instances belonging to a certain cluster. Therefore, we give the new updating rules for the centroids. Let $m_k = (m_k^{(1)}, m_k^{(2)}, m_k^{(3)}, m_k^{(4)})$ be the $k$-th centroid $C_k$, which $m_k^{(1)} = (m_k^1, \cdots, m_k^m)$, $m_k^{(2)} = (m_{k,m+1}, \cdots, m_{k,m+K})$, $m_k^{(3)} = (m_{k,m+K+1}, \cdots, m_{k,2m+K})$ and $m_k^{(4)} = (m_{k,2m+K+1}, \cdots, m_{k,2m+2K})$. Let $Z_1 = Z_{S_1} \cup Z_{T_1}$ and $Z_2 = Z_{S_2} \cup Z_{T_2}$, we modify the computation of the centroids as follows,

\begin{align}
    m_k^{(1)} &= \frac{\sum_{x_i \in C_k \cap Z_1} d_i^{(1)} }{|C_k \cap Z_1|}, \\
    m_k^{(2)} &= \frac{\sum_{x_i \in C_k \cap Y_{S_1}} d_i^{(2)} }{|C_k \cap Y_{S_1}|}, \\
    m_k^{(3)} &= \frac{\sum_{x_i \in C_k \cap Z_2} d_i^{(3)} }{|C_k \cap Z_2|}, \\
    m_k^{(4)} &= \frac{\sum_{x_i \in C_k \cap Y_{S_2}} d_i^{(4)} }{|C_k \cap Y_{S_2}|}.
\end{align}

(6.20)

Recall that in the standard K-means, the centroids are computed by arithmetic means, whose denominator represents the number of instances in its corresponding cluster. Here we only put the “real” instances into the computation of centroids. After modifying the computation of centroids, we have the following theorem.

**Theorem 6.2.1** Given the data matrix $Z_{S_1}, Z_{T_1}, Z_{S_2}, Z_{T_2}$ and the label information from two source
domains $Y_{S_1}$ and $Y_{S_2}$ and augmented matrix $D$, we have the following equivalence

\[
\min \left\| \begin{bmatrix} Z_{S_1} \\ Z_{T_1} \end{bmatrix} - \begin{bmatrix} H_{S_1} \\ H_{T} \end{bmatrix} \right\|_F^2 + \lambda \|Y_{S_1} - H_{S_1} M_1\|_F^2 \\
+ \left\| \begin{bmatrix} Z_{S_2} \\ Z_{T_2} \end{bmatrix} - \begin{bmatrix} H_{S_2} \\ H_{T} \end{bmatrix} \right\|_F^2 G_2 + \lambda \|Y_{S_2} - H_{S_2} M_2\|_F^2,
\]

\[\label{eq:6.21}
\Leftrightarrow \min \sum_{k=1}^K \sum_{d_i \in C_k} f(d_i, m_k),
\]

where the centroids are calculated by Eq. \eqref{eq:6.20} and the distance function $f$ can be computed by

\[
f(d_i, m_k) = \mathbf{1}(d_i \in Z_1) \|d_i^{(1)} - m_k^{(1)}\|_2^2 + \lambda \mathbf{1}(d_i \in Y_{S_1}) \|d_i^{(2)} - m_k^{(2)}\|_2^2 + \mathbf{1}(d_i \in Z_2) \|d_i^{(3)} - m_k^{(3)}\|_2^2 + \lambda \mathbf{1}(d_i \in Y_{S_2}) \|d_i^{(4)} - m_k^{(4)}\|_2^2,
\]

\[\label{eq:6.22}
\]

where $\mathbf{1}(\cdot)$ returns 1 when it meets the condition, otherwise returns 0.

**Remark 12** Theorem 6.2.1 gives a way to handle the problem in Eq. \eqref{eq:6.10} via a K-means-like optimization problem, which has a neat mathematical way and can be solved with high efficiency. After changing the update rule for centroids and the computation for the distance function, we can still use two-phase iterative optimization with data assignment and centroid update successively.

**Remark 13** With a close look at the augmented matrix $D$, the label information can be regarded as new features with more weights, which is controlled by $\lambda$. Besides, Theorem 6.2.1 provides a way to cluster with both numeric and categorical features together, which means we calculate the difference between the numeric and categorical part of two instances separately and add them together.

By Theorem 6.2.1 we transfer the problem into a K-means-like clustering problem. Although there are 10 unknown variables in a two-source domain scenario, the benefits of this solution are that not only the problem can be solved in a neat mathematical and efficient way, but also the model can be easily extended from two source domains to several source domains. Since the update rule and distance function have changed, it is necessary to verify the convergence of the K-means-like algorithm.

**Theorem 6.2.2** For the objective function in Theorem 6.2.1, the optimization problem is guaranteed to converge in finite two-phase iterations of K-means-like optimization problem.
CHAPTER 6. STRUCTURE-PRESERVED DOMAIN ADAPTATION

![Image of some image examples of Office+Caltech (a) and PIE (b), where they have four and five subsets (domains), respectively.]

Figure 6.1: Some image examples of Office+Caltech (a) and PIE (b), where they have four and five subsets (domains), respectively.

Note that the K-means-like optimization also suits for the single source domain adaptation in Eq. (6.3). Next, we analyze the time complexity. Since we equivalently transfer the problem into a K-means-like optimization problem, the time complexity of the proposed method enjoys the same time complexity with K-means, $O(tndK)$, where $t$ is the number of iteration, $n$ is the number of data instances including source and target domains, $d$ is the dimension of the concatenating matrix matrix, which equals to $2m + 2K$ and $m$ is the dimension of the common space of source and target domain. We summarize the algorithm in Algorithm 7. The process is similar to K-means clustering. The major differences are the distance function and update rule for centroids.

### 6.3 Experimental Results

In this section, we evaluate the performance of structure-preserved unsupervised domain adaptation algorithms in terms of two scenarios, object recognition and face identification.

#### 6.3.1 Experimental Settings

**Databases.** Office+Caltech is an increasingly popular benchmark for visual domain adaptation. The database contains three real-world object domains, Amazon (images downloaded from online merchants), Webcam (low-resolution images by a web camera), and DSLR (high-resolution images by a digital SLR camera). It has 4,652 images and 31 categories. Caltech-256 is a standard database for object recognition, which has 30,607 images and 256 categories. Here we adopt the public Office+Caltech datasets released by Gong et al. [158], which has four domains, C (Caltech-256), A (Amazon), W (Webcam), and D (DSLR) and 10 categories in each domain. SURF features are extracted and quantized into an 800-bin histogram with codebooks computed.
CHAPTER 6. STRUCTURE-PRESERVED DOMAIN ADAPTATION

Table 6.2: Performance (%) comparison on three multiple sources domain benchmarks using SURF features

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>NC</th>
<th>A-SVM</th>
<th>LTSL-PCA</th>
<th>LTSL-LDA</th>
<th>SFC-C</th>
<th>SFC-J</th>
<th>RDALR</th>
<th>FDDL</th>
<th>SDDL</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,D</td>
<td>W</td>
<td>20.6</td>
<td>30.4</td>
<td>55.5</td>
<td>30.2</td>
<td>52.0</td>
<td>64.5</td>
<td>36.9</td>
<td>41.0</td>
<td>57.8</td>
<td>76.3</td>
</tr>
<tr>
<td>A,W</td>
<td>D</td>
<td>16.4</td>
<td>25.3</td>
<td>57.4</td>
<td>43.0</td>
<td>39.0</td>
<td>51.3</td>
<td>31.2</td>
<td>38.4</td>
<td>56.7</td>
<td>73.9</td>
</tr>
<tr>
<td>D,W</td>
<td>A</td>
<td>16.9</td>
<td>17.3</td>
<td>20.0</td>
<td>17.1</td>
<td>29.0</td>
<td>38.4</td>
<td>20.9</td>
<td>19.0</td>
<td>24.1</td>
<td>43.8</td>
</tr>
</tbody>
</table>

Table 6.2: Performance (%) comparison on three multiple sources domain benchmarks using SURF features

with K-means on a subset of images from Amazon. Then the histograms are standardized by z-score. Beyond the SURF feature, deep features have also been extracted from this dataset for discriminative representation [180].

PIE, which stands for “Pose, Illumination, Expression”, is a benchmark face database. The database has 68 individuals with 41,368 face images of size 32 × 32. The face images are captured by 13 synchronized cameras (different poses) and 21 flashes (different illuminations and/or expressions). In these experiments, to thoroughly verify that our approach can perform robustly across different distributions, we adopt five subsets of PIE, each corresponding to a different pose. Specifically, we choose PIE05 (left pose), PIE07 (upward pose), PIE09 (downward pose), PIE27 (frontal pose), PIE29 (right pose). In each subset (pose), all the face images are taken under different lightings, and expression conditions. Some image examples are shown in Figure 6.1.

Competitive methods and implementation details. Here we evaluate the proposed method in scenarios of single source and multiple sources. Five competitive methods are employed in the single source setting, including Principal Component Analysis (PCA), Geodesic Flow Kernel (GFK) [158], Transfer Component Analysis (TCA) [164], Transfer Subspace Learning (TSL) [156] and Joint Domain Adaptation (JDA) [159]. GFK [158] models domain shift by integrating an infinite number of subspaces from the source to the target domain. TCA [153], TSL [156], JDA [159] and LSC [168] are four subspace based algorithms, which manages to seek a common shared subspace to mitigate the domain shift. The last two further incorporates the pseudo labels the target data to fight off the conditional distribution divergence across two domains. ARRLS employs the adaptation regularization to preserve the manifold consistency underlying marginal distribution [182]. For subspace-based methods (except LSC), we use the classical SVM to train the model on the source domain and predict the labels for the target domain data. Moreover, some deep learning methods are also involved for comparisons. CNN is a powerful network for image classification, which also has been proved that it is effective for learning transferable features [183]. LapCNN, a variant of CNN is proposed based on Laplacian graph regularization. Similarly, DDC is a domain adaptation variant of CNN that adds an adaptation layer between the $fc7$ and $fc8$ layers. DAN embeds the hidden representations of all task-specific layers in a reproducing kernel Hilbert space to address the
Table 6.3: Performance (%) comparison on Office+Caltech with one source using SURF features

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PCA</th>
<th>GFK</th>
<th>TCA</th>
<th>TSL</th>
<th>JDA</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>C→A</td>
<td>37.0</td>
<td>41.0</td>
<td>38.2</td>
<td>44.5</td>
<td>44.8</td>
<td>45.6</td>
</tr>
<tr>
<td>C→W</td>
<td>32.5</td>
<td>40.7</td>
<td>38.6</td>
<td>34.2</td>
<td>37.3</td>
<td>53.9</td>
</tr>
<tr>
<td>C→D</td>
<td>38.2</td>
<td>38.9</td>
<td>41.4</td>
<td>43.3</td>
<td>43.3</td>
<td>47.8</td>
</tr>
<tr>
<td>A→C</td>
<td>34.7</td>
<td>40.3</td>
<td>37.8</td>
<td>37.6</td>
<td>36.8</td>
<td>30.7</td>
</tr>
<tr>
<td>A→W</td>
<td>35.6</td>
<td>39.0</td>
<td>37.6</td>
<td>33.9</td>
<td>38.0</td>
<td>39.7</td>
</tr>
<tr>
<td>A→D</td>
<td>27.4</td>
<td>36.3</td>
<td>33.1</td>
<td>26.1</td>
<td>28.7</td>
<td>40.8</td>
</tr>
<tr>
<td>W→C</td>
<td>26.4</td>
<td>30.7</td>
<td>29.3</td>
<td>29.8</td>
<td>29.7</td>
<td>30.5</td>
</tr>
<tr>
<td>W→A</td>
<td>31.0</td>
<td>29.8</td>
<td>30.1</td>
<td>30.3</td>
<td>35.9</td>
<td>43.5</td>
</tr>
<tr>
<td>W→D</td>
<td>77.1</td>
<td>80.9</td>
<td>87.3</td>
<td>87.3</td>
<td>85.4</td>
<td>72.6</td>
</tr>
<tr>
<td>D→C</td>
<td>29.7</td>
<td>30.3</td>
<td>31.7</td>
<td>28.5</td>
<td>31.3</td>
<td>29.9</td>
</tr>
<tr>
<td>D→A</td>
<td>32.1</td>
<td>32.1</td>
<td>32.2</td>
<td>27.6</td>
<td>30.2</td>
<td>44.8</td>
</tr>
<tr>
<td>D→W</td>
<td>75.9</td>
<td>75.6</td>
<td>86.1</td>
<td>85.4</td>
<td>84.8</td>
<td>61.7</td>
</tr>
<tr>
<td>Average</td>
<td>39.8</td>
<td>43.0</td>
<td>43.6</td>
<td>42.4</td>
<td>44.9</td>
<td>45.1</td>
</tr>
</tbody>
</table>

Note: Since our method is based on JDA, our goal is to show the improvement over JDA.

Our method aims to better utilize the knowledge from the source domain, rather than to learn a better common space and therefore, we use the projection $P$ from JDA as the input of our methods. Accuracy is used for evaluating the performance of all methods. Since our method is a clustering based method, the best alignment is applied first, then the accuracy is calculated.

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} \delta(s_i, \text{map}(r_i))}{n},$$

(6.23)

where $\delta(x, y)$ equals one if $x = y$ and equals zero otherwise, and $\text{map}(r_i)$ is the permutation mapping.
Table 6.4: Performance (%) of our algorithm on Office+Caltech of our method with two source domains using SURF features

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Ours</th>
<th>Dataset</th>
<th>Ours</th>
<th>Dataset</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>C,W→A</td>
<td>54.8</td>
<td>C,D→A</td>
<td>54.4</td>
<td>D,W→A</td>
<td>43.8</td>
</tr>
<tr>
<td>C,A→W</td>
<td>52.5</td>
<td>C,D→W</td>
<td>80.0</td>
<td>A,D→W</td>
<td>76.3</td>
</tr>
<tr>
<td>C,W→D</td>
<td>80.3</td>
<td>C,A→D</td>
<td>51.0</td>
<td>A,W→D</td>
<td>73.9</td>
</tr>
<tr>
<td>A,W→C</td>
<td>40.8</td>
<td>A,D→C</td>
<td>43.5</td>
<td>D,W→C</td>
<td>35.1</td>
</tr>
</tbody>
</table>

Average: 57.2

function that maps each cluster label $r_i$ to the ground truth label.

### 6.3.2 Object Recognition with SURF Features

**Results of single source.** Here we demonstrate the effectiveness of our method with the scenario of one source and one target domain. From Table 6.3, we can see that our method with single source domain gets better results in 9 out of 12 datasets over JDA. Taking a close look, nearly 10% improvements compared to the second best results are made in $C \rightarrow W$, $A \rightarrow D$ and $D \rightarrow A$. However, the performance of our method on $D \rightarrow W$ and $W \rightarrow D$ is much worse than the one of other methods. In the following we utilize the multi-source domain data to improve the performance.

In the single source setting, although some methods can achieve very high accuracy, such as $D \rightarrow W$ and $A \rightarrow D$, the performance drops heavily when we choose another source domain. For example, the best result of $D \rightarrow W$ is 86.1%, while only 36.3% can be obtained on $A \rightarrow W$. This indicates that different sources play a crucially important role in the tasks on target domain. As for unsupervised domain adaptation, we cannot know the best source domain in advance and therefore, a robust method is always needed when we have multiple sources. Our method can also benefit the robustness from multiple sources setting. Even though two source domains have large discrepancies, such as $A, D \rightarrow W$, we can still obtain a satisfactory result. Since our method is based on JDA, our goal is to show the improvement over JDA. To our best knowledge, we also report the best performance on Office+Caltech CDDA [181] for complete understanding.

**Results of multiple sources.** Here we demonstrate the performance of our method in the multiple sources setting. In Table 6.4, in most of the cases, the performance with the multiple sources setting outperforms the one with single source. This indicates that our method fuses the different projected feature spaces in an effective way. When it comes to the average result, 12% improvements over the best result in the single source setting are achieved. Although we use more source data to achieve higher performance, it is still very appealing. In reality, it is easy to obtain many auxiliary well-labeled datasets. Table 6.2 shows the performance of different algorithms in the
CHAPTER 6. STRUCTURE-PRESERVED DOMAIN ADAPTATION

Figure 6.2: Performance (%) improvement of our algorithm in the multi-source setting compared to single source setting with SURF features. The blue and red bars denote two source domains, respectively. For example, in the first bar C,W → A, the blue bar shows the improvement of our method with two source domains C and W over the one only with the source domain C.

Figure 6.3: Parameter analysis of $\lambda$ with SURF feature on Office+Caltech.

multi-source setting. Our method renders obvious advantages over the other methods by over 20%. These competitive methods perform even worse than the methods on single source setting, which indicates that when it comes to complex multi-source scenario, the competitive methods learn the deformed common space and degrade the performance. On the contrary, our method preserves all the source structure and transfers the whole structures to the target domain.

If we take a close look at Figure 6.2 nearly in all the cases our method in the multi-source setting has substantial improvement over the one in the single source setting. This verifies that structure-preserved information from multi-source domains can help to boost the performance.
Table 6.5: Performance (%) on *Office+Caltech* with one source domain using deep features or deep models

<table>
<thead>
<tr>
<th>Dataset</th>
<th>C→A</th>
<th>C→W</th>
<th>C→D</th>
<th>A→C</th>
<th>A→W</th>
<th>A→D</th>
<th>W→C</th>
<th>W→A</th>
<th>W→D</th>
<th>D→C</th>
<th>D→A</th>
<th>D→W</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep model</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CNN</td>
<td>91.1</td>
<td>83.1</td>
<td>89.0</td>
<td>83.8</td>
<td>61.6</td>
<td>63.8</td>
<td>76.1</td>
<td>49.8</td>
<td>95.4</td>
<td>80.8</td>
<td>51.1</td>
<td>95.4</td>
<td>76.8</td>
</tr>
<tr>
<td>LapCNN</td>
<td>92.1</td>
<td>81.6</td>
<td>87.8</td>
<td>83.6</td>
<td>60.4</td>
<td>63.1</td>
<td>77.8</td>
<td>48.2</td>
<td>94.7</td>
<td>80.6</td>
<td>51.6</td>
<td>94.7</td>
<td>76.4</td>
</tr>
<tr>
<td>DAN</td>
<td>91.3</td>
<td>85.5</td>
<td>89.1</td>
<td>84.3</td>
<td>61.8</td>
<td>64.4</td>
<td>76.9</td>
<td>52.2</td>
<td>95.0</td>
<td>80.5</td>
<td>52.1</td>
<td>95.0</td>
<td>77.3</td>
</tr>
<tr>
<td>DDC</td>
<td>92.0</td>
<td>92.0</td>
<td>90.5</td>
<td>86.0</td>
<td>68.5</td>
<td>67.0</td>
<td>81.5</td>
<td>53.1</td>
<td>96.0</td>
<td>82.0</td>
<td>54.0</td>
<td>96.0</td>
<td>79.9</td>
</tr>
<tr>
<td>Deep features</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Direct</td>
<td>91.9</td>
<td>79.7</td>
<td>86.5</td>
<td>82.6</td>
<td>74.6</td>
<td>81.5</td>
<td>64.6</td>
<td>74.6</td>
<td>99.4</td>
<td>60.2</td>
<td>72.1</td>
<td>96.6</td>
<td>80.4</td>
</tr>
<tr>
<td>GFK</td>
<td>87.7</td>
<td>75.1</td>
<td>83.1</td>
<td>79.4</td>
<td>79.4</td>
<td>76.7</td>
<td>73.3</td>
<td>84.3</td>
<td>99.3</td>
<td>80.4</td>
<td>85.0</td>
<td>79.7</td>
<td>81.9</td>
</tr>
<tr>
<td>TCA</td>
<td>90.2</td>
<td>81.0</td>
<td>87.3</td>
<td>85.0</td>
<td>82.2</td>
<td>76.9</td>
<td>77.4</td>
<td>82.7</td>
<td>98.2</td>
<td>79.7</td>
<td>87.7</td>
<td>97.0</td>
<td>85.4</td>
</tr>
<tr>
<td>JDA</td>
<td>92.0</td>
<td>85.1</td>
<td>90.4</td>
<td>86.3</td>
<td>88.5</td>
<td>83.8</td>
<td>83.6</td>
<td>87.0</td>
<td>100</td>
<td>83.9</td>
<td>90.3</td>
<td>98.0</td>
<td>88.9</td>
</tr>
<tr>
<td>LSC</td>
<td>94.3</td>
<td>91.2</td>
<td>95.3</td>
<td>87.9</td>
<td>88.8</td>
<td>94.9</td>
<td>88.0</td>
<td>93.3</td>
<td>100</td>
<td>86.2</td>
<td>92.4</td>
<td>99.3</td>
<td>92.6</td>
</tr>
<tr>
<td>ARRLS</td>
<td>93.4</td>
<td>91.5</td>
<td>91.1</td>
<td>88.9</td>
<td>91.2</td>
<td>89.8</td>
<td>87.5</td>
<td>92.4</td>
<td>100</td>
<td>86.6</td>
<td>92.2</td>
<td>99.0</td>
<td>92.2</td>
</tr>
<tr>
<td>Ours</td>
<td>99.0</td>
<td>89.5</td>
<td>91.7</td>
<td>89.8</td>
<td>89.2</td>
<td>91.1</td>
<td>88.3</td>
<td>94.0</td>
<td>99.4</td>
<td>88.2</td>
<td>94.0</td>
<td>98.0</td>
<td>92.7</td>
</tr>
</tbody>
</table>
### Table 6.6: Performance (%) comparison on *Office+Caltech* with multi-source domains using deep features

<table>
<thead>
<tr>
<th>Source Target</th>
<th>A,C,D</th>
<th>A,C,W</th>
<th>C,D,W</th>
<th>A,D,W</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct</td>
<td>81.7</td>
<td>96.2</td>
<td>82.9</td>
<td>78.0</td>
<td>84.7</td>
</tr>
<tr>
<td>A-SVM</td>
<td>81.4</td>
<td>94.9</td>
<td>85.9</td>
<td>78.4</td>
<td>85.2</td>
</tr>
<tr>
<td>GFK</td>
<td>79.8</td>
<td>84.9</td>
<td>84.9</td>
<td>79.7</td>
<td>82.3</td>
</tr>
<tr>
<td>TCA</td>
<td>86.1</td>
<td>97.5</td>
<td>92.3</td>
<td>84.4</td>
<td>90.1</td>
</tr>
<tr>
<td>JDA</td>
<td>92.9</td>
<td>97.5</td>
<td>92.7</td>
<td>88.3</td>
<td>92.9</td>
</tr>
<tr>
<td>LSC</td>
<td>93.2</td>
<td>98.7</td>
<td>94.0</td>
<td>88.8</td>
<td>93.6</td>
</tr>
<tr>
<td>Ours</td>
<td>94.9</td>
<td>96.2</td>
<td>94.5</td>
<td>88.7</td>
<td>93.6</td>
</tr>
</tbody>
</table>

**Parameter analysis.** In our model, only one parameter $\lambda$ is used to control the similarity between the learnt indicator matrix and labels of the source domains. We expect to keep the structure of source domains and transfer that to the target domain. Intuitively, the larger $\lambda$ leads better performance. Therefore, we vary the $\lambda$ values from $10^{-5}$ to $10^{+5}$ to the change of performance. In Figure 6.3, we can see that on these 4 datasets the performance goes up with the increasing of $\lambda$ and when $\lambda$ reaches a certain value, the results become stable. Usually the performance is good enough when $\lambda = 100$. Therefore, $\lambda = 100$ is the default setting.

### 6.3.3 Object Recognition with Deep Features

Deep learning attracts more and more attention in recent years due to the dramatic improvement over the traditional methods. In essence, the features are extracted layer-by-layer for more effective information. In this subsection, we continue to work on the object recognition scenario and evaluate the performance of different unsupervised domain adaptation methods with deep features [180].

First we compare our method with K-means on the target data to demonstrate the benefit of our SP-UDA framework, which is exactly the first part of our framework. Figure 6.4 shows the performance improvement of our algorithm in the single source setting over K-means with deep features. We can see that our method has nearly 6%-30% improvements over K-means on different datasets, which results from the second structure-preserved term. The categorical utility function $U_c$ is usually to measure the submiliary between two partitions, while we apply $U_c$ to preserve the whole source structure. Different from the traditional pair-wise constraints, the source labels are treated as a whole to guide the target data clustering.

Table 6.5 shows the performance of several unsupervised domain adaptation methods in the single source domain setting. Compared with the results with SURF features in Table 6.3, the
CHAPTER 6. STRUCTURE-PRESERVED DOMAIN ADAPTATION

performance has significant improvements with deep features or deep models. This indicates that deep features or deep models are effective to learn the transferable features. It is worthy to note that even the Direct method easily outperforms the best result with SURF features and deep models. Therefore, the powerful features, which have the capacity for domain adaptation are crucial for the domain adaptation. With deep features, the domain adaptation methods can further boost the performance with positive transfer. Recall that our method is based on the common space learnt by JDA. It is exciting to see that our method has 3.6% improvement over JDA on average level.

Most existing domain adaptation methods employ the classification for the target data recognition, where only several key data points determine the hyperplane, and the target data are not involved to contribute the decision boundary. Differently, in the SP-UDA framework the whole source structure is utilized for transfer. Moreover, the target data and source data are put together to mutually determine the decision boundary. This indicates that the partition-level constraint can preserve the whole source structure for the guidance of target data clustering, which demonstrates the effectiveness of SP-UDA framework. Even with the simple K-means as the core clustering method, our method can achieve the competitive performance with the state-of-the-art methods.

Next we evaluate the performance in the multi-source setting. Table 6.6 shows the results with deep features. In the average level, the multi-source setting gains slight improvement over the result in single source setting in Table 6.5 and our method achieves competitive performance compared with rivals. In the last subsection, our model achieves lots of gains with multiple source domains and SURF features; however, less than 1% improvement has been obtained with deep
features. If we compare the results in Table 6.5 and 6.6, it comes to the same conclusion that it is difficult to boost the result of domain adaptation with deep features. This makes sense since the deep structure exacts discriminative but similar representation. Although this kind of features is promising for recognition, different source domains have too little complementary information for further improvement.

6.3.4 Face Identification

Domain adaptation results. Next, we verify our model in the face identification scenario. Table 6.7 shows the results with single or multiple sources and one target setting. Similar observations can be found. (1) In most of cases, our method for multi-source domains achieves the best results; (2) it is difficult to determine which source is the best for a given target domain. For example, although one source setting obtains very good performance on some datasets, such as 27 → 9 and 27 → 7, the result of 27 → 29 only gets about 40% accuracy. Our method based on multi-source domains leads to benefit the robustness and obtains the satisfactory results. In general, our average result exceeds other methods by a large margin.

Convergence study. Finally, we conduct the convergence study. The convergence of our model has been proven in the previous section, and we experimentally study the speed of convergence of our model. Figure 6.5 shows the convergence curve of 5, 29 → 9. We can see that our model converges fast within 10 iterations, which demonstrates the high efficiency of the proposed method.
Table 6.7: Performance (%) on PIE with one or multi-source and one target setting

| Dataset | PCA | GFK | TCA | TSL | JDA | Ours | Dataset | PCA | GFK | TCA | TSL | JDA | Ours | Dataset | PCA | GFK | TCA | TSL | JDA | Ours |
|---------|-----|-----|-----|-----|-----|------|---------|-----|-----|-----|-----|-----|------|---------|-----|-----|-----|-----|-----|------|---------|-----|-----|-----|-----|-----|------|
| 7 → 5  | 24.2 | 25.2 | 41.8 | 46.8 | 49.2 | 57.4 | 7 → 5  | 24.2 | 25.2 | 41.8 | 46.8 | 49.2 | 67.5 | 7 → 5  | 24.2 | 25.2 | 41.8 | 46.8 | 49.2 | 58.3 |
| 9 → 5  | 21.0 | 21.8 | 34.7 | 37.0 | 47.8 | 45.4 | 9 → 5  | 21.0 | 21.8 | 34.7 | 37.0 | 47.8 | 58.2 | 27 → 5 | 32.0 | 34.2 | 55.6 | 63.7 | 64.2 | 66.8 |
| 27 → 5 | 32.0 | 34.2 | 55.6 | 63.7 | 64.2 | 45.4 | 29 → 5 | 18.9 | 20.4 | 27.0 | 33.3 | 47.2 | 58.2 | 27 → 5 | 32.0 | 34.2 | 55.6 | 63.7 | 64.2 | 66.8 |
| 5 → 7  | 24.8 | 26.2 | 40.8 | 44.1 | 40.0 | 43.8 | 5 → 7  | 24.8 | 26.2 | 40.8 | 44.1 | 40.0 | 60.0 | 5 → 7  | 24.8 | 26.2 | 40.8 | 44.1 | 40.0 | 44.9 |
| 9 → 7  | 40.1 | 43.2 | 47.7 | 47.0 | 32.0 | 51.0 | 9 → 7  | 40.1 | 43.2 | 47.7 | 47.0 | 32.0 | 40.1 | 27 → 7 | 61.0 | 62.9 | 67.8 | 72.7 | 48.5 | 51.0 |
| 27 → 7 | 61.0 | 62.9 | 67.8 | 72.7 | 48.5 | 51.0 | 29 → 7 | 23.4 | 24.6 | 29.9 | 34.1 | 27.3 | 51.8 | 29 → 7 | 23.4 | 24.6 | 29.9 | 34.1 | 27.3 |
| 5 → 9  | 25.2 | 27.3 | 41.8 | 47.5 | 43.4 | 53.1 | 5 → 9  | 25.2 | 27.3 | 41.8 | 47.5 | 43.4 | 50.0 | 5 → 9  | 25.2 | 27.3 | 41.8 | 47.5 | 43.4 | 55.7 |
| 27 → 9 | 45.5 | 47.4 | 51.5 | 57.6 | 57.8 | 93.5 | 27 → 9 | 27 → 7 | 72.2 | 73.4 | 75.9 | 83.5 | 43.4 | 58.0 | 29 → 9 | 27.2 | 28.5 | 29.9 | 36.6 | 38.5 |
| 7 → 9  | 45.5 | 47.4 | 51.5 | 57.6 | 57.8 | 46.7 | 7 → 9  | 45.5 | 47.4 | 51.5 | 57.6 | 37.9 | 51.0 | 27 → 9 | 72.2 | 73.4 | 75.9 | 83.5 | 43.4 | 58.0 |
| 27 → 9 | 72.2 | 73.4 | 75.9 | 83.5 | 43.4 | 51.0 | 29 → 9 | 27.2 | 28.5 | 29.9 | 36.6 | 38.5 | 51.0 | 29 → 9 | 27.2 | 28.5 | 29.9 | 36.6 | 38.5 |
| 5 → 7  | 163 | 17.6 | 29.4 | 36.2 | 67.0 | 71.6 | 5 → 7  | 16.3 | 17.6 | 29.4 | 36.2 | 67.0 | 71.0 | 5 → 7  | 16.3 | 17.6 | 29.4 | 36.2 | 67.0 | 71.7 |
| 7 → 7  | 53.4 | 54.3 | 64.7 | 71.4 | 37.9 | 50.1 | 7 → 7  | 53.4 | 54.3 | 64.7 | 71.4 | 37.9 | 71.0 | 5 → 7  | 16.3 | 17.6 | 29.4 | 36.2 | 67.0 | 71.7 |
| 9 → 7  | 46.1 | 46.4 | 56.2 | 59.5 | 30.9 | 50.1 | 27 → 9 | 30.3 | 31.3 | 33.6 | 38.8 | 46.4 | 58.4 | 27 → 9 | 30.3 | 31.3 | 33.6 | 38.8 | 46.4 |
| 27 → 9 | 53.4 | 54.3 | 64.7 | 71.4 | 37.9 | 53.4 | 29 → 7 | 30.3 | 31.3 | 33.6 | 38.8 | 46.4 | 58.4 | 27 → 9 | 30.3 | 31.3 | 33.6 | 38.8 | 46.4 |
| 5 → 9  | 16.3 | 17.6 | 29.4 | 36.2 | 47.1 | 50.3 | 5 → 9  | 16.3 | 17.6 | 29.4 | 36.2 | 47.1 | 51.5 | 5 → 9  | 16.3 | 17.6 | 29.4 | 36.2 | 47.1 |
| 7 → 9  | 25.4 | 27.1 | 33.7 | 35.7 | 26.2 | 50.3 | 9 → 29 | 25.4 | 27.1 | 33.7 | 35.7 | 26.2 | 38.5 | 9 → 29 | 25.4 | 27.1 | 33.7 | 35.7 | 26.2 |
| 29 → 9 | 25.4 | 27.1 | 33.7 | 35.7 | 26.2 | 53.1 | 27 → 29 | 35.1 | 38.4 | 40.3 | 44.8 | 39.7 | 47.2 | 27 → 29 | 35.1 | 38.4 | 40.3 | 44.8 | 39.7 |

Average: PCA(33.2)  GFK(34.7)  TCA(43.2)  TSL(48.1)  JDA(42.2)  Ours(54.2)
6.4 Summary

In this chapter, we proposed a novel framework for unsupervised domain adaptation named structure-preserved unsupervised domain adaptation (SP-UDA). Different from the existing studies, which learnt a classification on a source domain and predicted the labels for target data, we preserved the whole structures of source domain for the task on the target domain. Generally speaking, both source and target data were put together for clustering, which simultaneously explored the structures of source and target domains. In addition, the well-preserved structure information from the source domain facilitated and guided the adaptation process in the target domain in a semi-supervised clustering fashion. To our best knowledge, we were the first to formulate the problem into a semi-supervised clustering problem with target labels as missing values. In addition, we solved the problem by a K-means-like optimization problem in an efficient way. Extensive experiments on two widely used databases demonstrated the large improvements of our proposed method over several state-of-the-art methods.
Chapter 7

Conclusion

In this thesis, we focus on the consensus clustering. Different from the traditional clustering algorithms, which separate a bunch of instances into different groups, the consensus clustering aim to fuse several basic clustering results derived from these traditional clustering algorithms into an integrated one. In essence, consensus clustering is a fusion problem, rather than the clustering problem. Generally speaking, consensus clustering can roughly be divided into two categories, utility function and co-association matrix.

For the utility function based methods, the challenges lie in how to design an effective utility function measuring the similarity between the basic partition and the consensus one, and how to solve it efficiently. To handle this, in Chapter 2, we propose K-means-based Consensus Clustering (KCC) utility functions, which transform the consensus clustering into K-means clustering on a binary matrix with theoretical supports. For the co-association matrix based methods, we propose Spectral Ensemble Clustering (SEC), which applies the spectral clustering on the co-association matrix. To solve it efficiently, a weighted K-means solution is put forward, which achieves SEC in an theoretical equivalent way. Later, Infinite Ensemble Clustering (IEC) is proposed, which aims to fuse infinite basic partitions for robust solution. To achieve this, we build the equivalent connection between IEC and marginalized denoising auto-encoder. Inspired by consensus clustering, especially on the utility function, the structure-preserved learning framework is designed and applied in constraint clustering and domain adaptation in Chapter 5 and 6, respectively.

In sum, our major contributions lie in building connections between different domains, and transforming complex problems into simple ones. In the future, I will continue the structure-preserved learning for other topics, including heterogenous domain adaptation, interpretable clustering and clustering with outlier removal.
Bibliography


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


Appendix A

Appendix

A.1 Proof of Lemma 2.2.1

Proof. According to the definition of centroids in K-means clustering, we have

$$m_{k,ij} = \frac{\sum_{x_l \in C_k} x_{l,ij}^{(b)}}{|C_k|}. \tag{A.1}$$

Recall the contingency matrix in Table 2.2, we have $|C_k| = n_{k+}$, and $\sum_{x_l \in C_k} x_{l,ij}^{(b)} = |C_k \cap C_j^{(i)}| = n_{kj}^{(i)}$. As a result,

$$m_{k,ij} = \frac{n_{kj}^{(i)}}{n_{k+}} = \frac{p_{kj}^{(i)}}{p_{k+}}, \tag{A.2}$$

and Eq. (2.10) thus follows.

A.2 Proof of Lemma 2.2.2

Proof. We begin the proof by giving one important fact as follows. Suppose $f$ is a distance function that fits K-means clustering. Then according to Eq. (2.3), $f$ is a point-to-centroid distance that can be derived by a differentiable convex function $\phi$. Therefore, if we substitute $f$ in Eq. (2.3) into the right-hand-side of Eq. (2.11), we have

$$\sum_{k=1}^{K} \sum_{x_l \in C_k} f(x_l^{(b)}, m_k) = \sum_{x_l^{(b)} \in \chi^{(b)}} \phi(x_l^{(b)}) - n \sum_{k=1}^{K} p_{k+} \phi(m_k). \tag{A.3}$$
Since both $\sum_{x_l^b} \phi(x_l^b)$ and $n$ are constants given $\Pi$, we have

$$\min_{\pi} \sum_{k=1}^K \sum_{x_l \in C_k} f(x_l^b, m_k) \iff \max_{\pi} \sum_{k=1}^K p_{k+\phi(m_k)}. \quad (A.4)$$

Now let us turn back to the proof of the sufficient condition. As $g_{\Pi,K}$ is strictly increasing, we have

$$\max_{\pi} \sum_{i=1}^r w_i U(\pi, \pi_i) \iff \max_{\pi} \sum_{k=1}^K p_{k+\phi(m_k)}. \quad (A.5)$$

So we finally have Eq. (2.11), which indicates that $U$ is a KCC utility function. The sufficient condition holds.

We then prove the necessary condition. Suppose the distance function $f$ in Eq. (2.11) is derived from a differentiable convex function $\phi$. According to Eq. (2.11) and Eq. (A.4), we have Eq. (A.5).

Let $\Upsilon(\pi)$ denote $\sum_{i=1}^r w_i U(\pi, \pi_i)$ and $\Psi(\pi)$ denote $\sum_{k=1}^K p_{k+\phi(m_k)}$ for the convenience of description. Note that Eq. (A.5) holds for any feasible region $F$ since Eq. (A.4) is derived from the equality rather than equivalence relationship in Eq. (A.3). Therefore, for any two consensus partitions $\pi'$ and $\pi''$, if we let $F = \{\pi', \pi''\}$, we have

$$\Upsilon(\pi') > (=, or <) \Psi(\pi'') \iff \Psi(\pi') > (=, or <) \Psi(\pi''), \quad (A.6)$$

which indicates that $\Upsilon$ and $\Psi$ have the same ranking over all the possible partitions in the universal set $F^* = \{\pi | L_{\pi}(x_l^b) \in \{1, \cdots, K\}, 1 \leq l \leq n\}$. Define a mapping $g_{\Pi,K}$ that maps $\Psi(\pi)$ to $\Upsilon(\pi)$, $\pi \in F^*$. According to Eq. (A.6), $g_{\Pi,K}$ is a function, and $\forall \ x > x', \ g_{\Pi,K}(x) > g_{\Pi,K}(x')$. This implies that $g_{\Pi,K}$ is strictly increasing. So the necessary condition holds, and the whole lemma thus follows.
APPENDIX A. APPENDIX

A.3 Proof of Theorem 2.2.1

Proof. We first prove the sufficient condition. If we substitute $U(\pi, \pi_i)$ in Eq. (2.13) into the left-hand-side of Eq. (2.12), we have

$$\sum_{i=1}^{r} w_i U(\pi, \pi_i) = \sum_{k=1}^{K} p_k \sum_{i=1}^{r} w_i \mu_i (P_k^{(i)})$$

$$= \frac{1}{a} \sum_{k=1}^{K} p_k \sum_{i=1}^{r} w_i \mu_i (m_{k,i}) - \frac{1}{a} \sum_{i=1}^{r} w_i c_i$$

(A.7)

where (\alpha) holds due to Eq. (2.15), and (\beta) holds due to Eq. (2.14). Let $g_{\Pi, K}(x) = a \Pi x + b_\Pi$, where $a = 1/a$ and $b = -1/a \sum_{i=1}^{r} w_i c_i$. Apparently, $g_{\Pi, K}$ is a strictly increasing function for $a > 0$. We then have

$$\sum_{i=1}^{r} w_i U(\pi, \pi_i) = g_{\Pi, K}(\sum_{k=1}^{K} p_k + \phi(m_k))$$

which indicates that $U$ is a KCC utility function. The sufficient condition thus holds. It remains to prove the necessary condition.

Recall Lemma 2.2.2. Due to the arbitrariness of $\Pi$, we can let $\Pi = \Pi_i = \{\pi_i\} (1 \leq i \leq r)$. Accordingly, $m_k$ reduces to $m_{k,i}$ in Eq. (2.12), and $\phi(m_{k,i})$ reduces to $\phi_i(m_{k,i})$, i.e., the $\phi$ function defined only on the $i$th “block” of $m_k$ without involvement of the weight $w_i$. Then according to Eq. (2.12), we have

$$U(\pi, \pi_i) = g_{\Pi_i, K}(\sum_{k=1}^{K} p_k + \phi_i(m_{k,i}))$$

(A.8)

where $g_{\Pi_i, K}$ is the mapping function when $\Pi$ reduces to $\Pi_i$. By summing up $U(\pi, \pi_i)$ from $i = 1$ to $r$, we have

$$\sum_{i=1}^{r} w_i U(\pi, \pi_i) = \sum_{i=1}^{r} w_i g_{\Pi_i, K}(\sum_{k=1}^{K} p_k + \phi_i(m_{k,i}))$$

which, according to Eq. (2.12), indicates that

$$g_{\Pi, K}(\sum_{k=1}^{K} p_k + \phi(m_k)) = \sum_{i=1}^{r} w_i g_{\Pi_i, K}(\sum_{k=1}^{K} p_k + \phi_i(m_{k,i})).$$

(A.9)

If we take the partial derivative with respect to $m_{k,ij}$ on both sides, we have

$$g'_{\Pi, K}(\sum_{k=1}^{K} p_k + \phi(m_k)) \frac{\partial \phi(m_k)}{\partial m_{k,ij}} = w_i g'_{\Pi_i, K}(\sum_{k=1}^{K} p_k + \phi_i(m_{k,i})) \frac{\partial \phi_i(m_{k,i})}{\partial m_{k,ij}}.$$  

(A.10)
As $(\gamma)$ and $(\delta)$ do not contain any weight parameters $w_l$, $1 \leq l \leq r$, the right-hand-side of Eq. (A.10) has one and only one weight parameter: $w_i$. This implies that $(\alpha)$ is a constant, otherwise the left-hand-side of Eq. (A.10) would contain multiple weight parameters other than $w_i$ due to the existence of $\phi(m_k)$ in $g'_{\Pi,K}$. Analogously, since $(\beta)$ does not contain all $p_{k+i}$, $1 \leq k \leq K$, $(\gamma)$ must also be a constant. These results imply that $g_{\Pi,K}(x)$ and $g_{\Pi,K_i}(x)$, $1 \leq i \leq r$, are all linear functions. Without loss of generality, we let

$$g_{\Pi,K}(x) = a_\Pi x + b_\Pi, \ \forall a_\Pi \in \mathbb{R}_{++}, b_\Pi \in \mathbb{R} \quad \text{and} \quad (A.11)$$

$$g_{\Pi,K_i}(x) = a_i x + b_i, \ \forall a_i \in \mathbb{R}_{++}, b_i \in \mathbb{R} \quad \text{(A.12)}$$

As a result, Eq. (A.10) turns into

$$a_\Pi \frac{\partial \phi(m_k)}{\partial m_{k,ij}} = w_i a_i \frac{\partial \phi_i(m_{k,i})}{\partial m_{k,ij}}, \ \forall i, j. \quad (A.13)$$

Apparently, $\frac{\partial \phi_i(m_{k,i})}{\partial m_{k,ij}}$ is the function of $\{m_{k,i1}, \ldots, m_{k,iK_i}\}$ only, which implies that $\frac{\partial \phi(m_k)}{\partial m_{k,ij}}$ is not the function of $m_{k,l}$, $\forall l \neq i$. As a result, given the arbitrariness of $i$, we have $\phi(m_k) = \varphi(\phi_1(m_{k,1}), \ldots, \phi_r(m_{k,r}))$, which indicates that

$$\frac{\partial \phi(m_k)}{\partial m_{k,ij}} = \frac{\partial \varphi}{\partial \phi_i} \frac{\partial \phi_i(m_{k,i})}{\partial m_{k,ij}}.$$ 

Accordingly, Eq. (A.13) turns into $\frac{\partial \varphi}{\partial \phi_i} = w_i a_i / a_\Pi$, which leads to

$$\phi(m_k) = \sum_{i=1}^{r} \left( \frac{w_i a_i}{a_\Pi} \phi_i(m_{k,i}) + d_i \right), \ \forall d_i \in \mathbb{R}. \quad (A.14)$$

Let $\nu_i(x) = \frac{a_\Pi}{a_i} \phi_i(x) + \frac{d_i}{w_i}$, $1 \leq i \leq r$, Eq. (2.14) thus follows.

Moreover, according to Eq. (A.8) and Eq. (A.12), we have

$$U(\pi, \pi_i) = \sum_{k=1}^{K} p_{k+}(a_i \phi_i(F_k^{(i)}) + b_i), \ \forall i. \quad (A.15)$$

Let $\mu_i(x) = a_i \phi_i(x) + b_i$, $1 \leq i \leq r$, Eq. (2.13) thus follows. If we further let $a = 1/a_\Pi$ and $c_i = d_i/w_i - b_i/a_\Pi$, we also have Eq. (2.15). The necessary condition thus holds. We complete the proof.
APPENDIX A. APPENDIX

A.4 Proof of Proposition 2.2.1

Proof. As $\mu$ is a convex function, according to the Jensen’s inequality, we have

\[
U_{\mu}(\pi, \pi_i) = \sum_{k=1}^{K} p_k + \mu\left(\frac{\sum_{k=1}^{K} p_k}{p_k+}, \cdots, \frac{\sum_{k=1}^{K} p_k}{p_k+}\right) \geq \mu\left(\sum_{k=1}^{K} \frac{p_k}{p_k+}\right)
\]

(A.16)

which indicates that the centroid-updating phase of K-means will also decrease $F$. Therefore, we guarantee that each two-phase iteration will decrease $F$ continuously. Furthermore, since the consensus partition $\pi$ has limited combinations, say $K^n$ for $K$ clusters, the iteration will definitely converge to a local minimum or a saddle point within finite iterations. We complete the proof.

A.5 Proof of Theorem 2.3.1

Proof. First, according to Eq. (2.21), it is easy to note that the assigning phase of K-means clustering will decrease $F$ monotonically. Moreover, since $f_i$ is a point-to-centroid distance, it can be derived by some continuously differentiable convex function $\phi_i$, i.e.,

\[
f_i(x, y) = \phi_i(x) - \phi_i(y) - (x - y)^T \nabla \phi_i(y), \forall i.
\]

(A.17)

Then according to Eq. (2.22), we have $\forall y_k \neq m_k$,

\[
F(y_k) - F(m_k) = \sum_{i=1}^{r} \sum_{k=1}^{K} \sum_{x_i \in C_k \cap X_i} \phi_i(m_{k,i}) - \phi_i(y_{k,i}) - (x_{l,i}^{(b)} - y_{k,i})^T \nabla \phi_i(y_{k,i}) + (x_{l,i}^{(b)} - m_{k,i})^T \nabla \phi_i(m_{k,i}).
\]

(A.18)

Substituting $\sum_{x_i \in C_k \cap X_i} x_{l,i}^{(b)}$ by $\sum_{x_i \in C_k \cap X_i} m_{k,i}$, we finally have

\[
F(y_k) - F(m_k) = \sum_{i=1}^{r} \sum_{k=1}^{K} n_{k+i}^{(i)} f_i(m_{k,i}, y_{k,i}) \geq 0,
\]

which indicates that the centroid-updating phase of K-means will also decrease $F$. Therefore, we guarantee that each two-phase iteration will decrease $F$ continuously. Furthermore, since the consensus partition $\pi$ has limited combinations, say $K^n$ for $K$ clusters, the iteration will definitely converge to a local minimum or a saddle point within finite iterations. We complete the proof.
A.6 Proof of Theorem 3.1.1

Proof. Let \( Y = \{ y = b(x)/w_b(x) \} \) and \( W_k \) denote the diagonal matrix of the weights in cluster \( C_k \), and \( Y_k \) denote the matrix of binary data associated with cluster \( C_k \). Then the centroid \( m_k \) can be rewritten as

\[
m_k = e^\top W_k Y_k / s_k,
\]

where \( e \) is the vector of all ones with appropriate size and \( s_k = e^\top W_k e \). According to [71], we have

\[
SSE_{C_k} = \sum_{x \in C_k} w_b(x) ||b(x)/w_b(x) - m_k||^2
\]

\[
= ||(I - W_k^{1/2} ee^\top W_k^{1/2} / s_k)W_k^{1/2} Y_k||_F^2
\]

\[
= tr(Y_k^\top W_k^{1/2} (I - W_k^{1/2} ee^\top W_k^{1/2} / s_k)^2W_k^{1/2} Y_k)
\]

\[
= tr(W_k^{1/2} Y_k Y_k^\top W_k^{1/2}) - \frac{e^\top W_k e}{\sqrt{s_k}} Y_k Y_k^\top W_k e / \sqrt{s_k}.
\]

If we sum up SSE of all the clusters, we have

\[
\sum_{k=1}^K \sum_{x \in C_k} w_b(x) ||b(x)/w_b(x) - m_k||^2 = tr(W^{1/2} YY^\top W^{1/2}) - tr(G^\top W^{1/2} YY^\top W^{1/2} G),
\]

where \( G = \text{diag}(W_k^{1/2} / e / \sqrt{s_k}, \ldots, W_k^{1/2} / e / \sqrt{s_k}) \). Recall that \( YY^\top = W^{-1} BB^\top W^{-1} \) and \( S = BB^\top, D = W \) and \( Z^\top Z = G^\top G = I \), so we have

\[
\max \ tr(Z^\top D^{-1/2} S D^{-1/2} Z) \Leftrightarrow \max \ tr(G^\top W^{-1/2} BB^\top W^{-1/2} G).
\]

The constant \( tr(W^{-1/2} BB^\top W^{-1/2}) \) finishes the proof.

A.7 Proof of Theorem 3.1.2

Proof. Given the equivalence of SEC and weighted K-means, we here derive the utility function of SEC. We start from the objective function of weighted K-means as follows:
APPENDIX A. APPENDIX

\[
\sum_{k=1}^{K} \sum_{x \in C_k} w_b(x) || b(x) - m_k ||^2 = \sum_{k=1}^{K} \sum_{x \in C_k} \frac{||b(x)||^2}{w_b(x)} - 2 \sum_{x \in C_k} b(x)m_k^T + \sum_{x \in C_k} w_b(x)||m_k||^2
\]

\[
= \sum_{k=1}^{K} \sum_{x \in C_k} \frac{||b(x)||^2}{w_b(x)} - 2 \sum_{x \in C_k} w_b(x)||m_k||^2 + \sum_{x \in C_k} w_b(x)||m_k||^2
\]

\[
= \sum_{k=1}^{K} \sum_{x \in C_k} \frac{||b(x)||^2}{w_b(x)} - \sum_{i=1}^{r} \sum_{k=1}^{K} w_{C_k} ||m_{k,i}||^2
\]

\[
= \sum_{k=1}^{K} \sum_{x \in C_k} \frac{||b(x)||^2}{w_b(x)} - r \sum_{i=1}^{r} \sum_{k=1}^{K} \frac{n_{k,i}}{w_{C_k}} \sum_{j=1}^{r} \frac{p_{k,j}}{p_{k,+}}^2
\]

Note that \((\gamma)\) is a constant and according to the definition of centroids in K-means, we have

\[
m_{k,ij} = \frac{\sum_{x \in C_k} b(x)_{ij}}{\sum_{x \in C_k} w_b(x)} = \frac{n_{k,i}^{(i)}}{w_{C_k}} = \frac{n_{k,i}}{n_{k,+}/w_{C_k}} = \frac{(n_{k,i}/n_{k,+})(n_{k,+}/w_{C_k})}{(p_{k,i}/p_{k,+})(n_{k,+}/w_{C_k})}.
\]

Thus we get the utility function of SEC.

A.8 Proof of Theorem 3.2.1

We first give a lemma as follows.

Lemma A.8.1

\[
f_{m_1,\ldots,m_K}(x) \in [0, 1].
\]

**Proof.** It is easy to show \( ||b(x)||^2 = r, w_b(x) \in [r, (n-K+1)r] \) and \( f_{m_1,\ldots,m_K}(x) \leq \max\{\frac{||b(x)||^2}{w_b(x)}, w_b(x)||m_k||^2\} \).

We have

\[
\frac{||b(x)||^2}{w_b(x)} \leq \frac{r}{r} = 1 \quad \text{and}
\]

\[
w_b(x)||m_k||^2 = \frac{w_b(x) \| \sum_{b(x) \in C_k} b(x) \| ^2}{(\sum_{b(x) \in C_k} w_b(x))^2} \leq 1.
\]

(A.19)

This concludes the proof.
A detailed proof of equation (A.19): If $|C_k| = 1$, the equation holds trivially. When $|C_k| \geq 2$, we have

$$
\frac{w_b(x)\|\sum_{b(x)\in C_k} b(x)\|^2}{(\sum_{b(x)\in C_k} w_b(x))^2} \leq \frac{w_b(x)\|b(x)\|^2}{(\sum_{b(x)\in C_k} w_b(x))^2}
$$

$$
= \frac{w_b(x)\sum_{b(x)\in C_k} r}{(w_b(x) + \sum_{b(x)\in C_k - \{b(x)\}} w_b(x))^2}
$$

$$
\leq \frac{w_b(x)|C_k| r}{(w_b(x) + (|C_k| - 1)r)^2}
$$

$$
= \frac{w_b(x)|C_k| r}{(w_b(x))^2 + 2w_b(x)(|C_k| - 1)r}
$$

$$
\leq \frac{|C_k| r}{w_b(x) + 2(|C_k| - 1)r} \leq \frac{|C_k| r}{|C_k| r + |C_k| r - r} \leq 1.
$$

The first inequality holds due to the triangle inequality.

Now we begins the proof of Theorem 3.

Proof. We have

$$
|f_{m_1,\ldots,m_K}(x) - f_{m_1,\ldots,m_K}(x')|
$$

$$
= |\min_k w_b(x)\| b(x) - b(x') \| - m_k || - \min_k w_b(x')\| b(x') - m_k ||
$$

$$
\leq \max_k w_b(x)\| b(x) - m_k || - w_b(x')\| b(x') - m_k ||
$$

$$
= \max_k \frac{r}{w_b(x)} - \| b(x) - m_k \| + w_b(x)\| m_k || - \frac{r}{w_b(x')} + \| b(x') - m_k || m_k ||^2
$$

$$
\leq \max_k \frac{r}{w_b(x)} - \frac{r}{w_b(x')} + \| b(x) - b(x') \| + \| m_k ||^2 w_b(x) - w_b(x').
$$
Note that the last inequality holds due to the Cauchy-Schwartz inequality. Recall that we have proved in Lemma 1 that $\|m_k\|^2 \leq \frac{1}{\min_{x \in X} w_b(x)}$, we have

$$|f_{m_1, \ldots, m_K}(x) - f_{m_1, \ldots, m_K}(x')|$$

$$\leq \max_k (\frac{r}{w_b(x)} - \frac{r}{w_b(x')}) + \|b(x) - b(x')\||m_k\| + \|m_k\|^2 |w_b(x) - w_b(x')|$$

$$\leq \max_k (\frac{r}{\min_{x \in X} (w_b(x))^2} + \|m_k\|^2 |w_b(x) - w_b(x')| + \|b(x) - b(x')\||m_k\|$$

$$\leq \frac{r + \min_{x \in X} w_b(x)}{\min_{x \in X} (w_b(x))^2} \sum_{i=1}^{r} \gamma_{w,i} + \left(\frac{\sum_{i=1}^{r} \gamma_{i}^2}{\min_{x \in X} w_b(x)}\right)^{1/2}$$

$$\leq \frac{2 \sum_{i=1}^{r} \gamma_{w,i}}{r} + \left(\frac{\sum_{i=1}^{r} \gamma_{i}^2}{r}\right)^{1/2}.$$
The expected Rademacher complexity and Gaussian complexity are defined as:

\[ R(F) = \mathbb{E}[R_n(F)] \text{ and } G(F) = \mathbb{E}[G_n(F)]. \]

Using the symmetric distribution property of random variables, we have:

**Theorem A 1** Let \( F \) be a real-valued function class on \( X \) and \( X = (x_1, \ldots, x_n) \in X^n \). Let

\[ \Phi(X) = \sup_{f \in F} \frac{1}{n} \sum_{i=1}^{n} (E_x f(x) - f(x_i)). \]

Then, \( E_x \Phi(X) \leq 2R(F) \).

The following theorem [192], proved utilizing Theorem A 1 and McDiarmid’s inequality, plays an important role in proving the generalization error bounds:

**Theorem A 2** Let \( F \) be an \([a, b]\)-valued function class on \( X \), and \( X = (x_1, \ldots, x_n) \in X^n \). For any \( f \in F \) and \( \delta > 0 \), with probability at least \( 1 - \delta \), we have

\[ E_x f(x) - \frac{1}{n} \sum_{i=1}^{n} f(x_i) \leq 2R(F) + (b - a) \sqrt{\frac{\ln(1/\delta)}{2n}}. \]

Combining Theorem A 2 and Lemma 1, we have

**Theorem A 3** Let \( \pi \) be any partition learned by SEC. For any independently distributed instances \( x_1, \ldots, x_n \) and \( \delta > 0 \), with probability at least \( 1 - \delta \), the following holds

\[ E_x f_{m_1, \ldots, m_K}(x) - \frac{1}{n} \sum_{l=1}^{n} f_{m_1, \ldots, m_K}(x_l) \leq 2R(F_{\Pi_K}) + \sqrt{\frac{\ln(1/\delta)}{2n}}. \]

We use Lemmas A.9.1 and A.9.2 (see proofs in [193]) to upper bound \( R(F_{\Pi_K}) \) by finding a proper Gaussian process which can easily be bounded.

**Lemma A.9.1 (Slepian’s Lemma)** Let \( \Omega \) and \( \Xi \) be mean zero, separable Gaussian processes indexed by a common set \( S \), such that

\[ E(\Omega_{s_1} - \Omega_{s_2})^2 \leq E(\Xi_{s_1} - \Xi_{s_2})^2, \forall s_1, s_2 \in S. \]

Then \( E \sup_{s \in S} \Omega_s \leq E \sup_{s \in S} \Xi_s. \)

The Gaussian complexity is related to the Rademacher complexity by the following lemma:
Lemma A.9.2

\[ \mathcal{R}(F) \leq \sqrt{\pi/2} \mathcal{G}(F). \]

Now, we can upper bound the Rademacher complexity \( \mathcal{R}(F) \) by finding a proper Gaussian process.

Lemma A.9.3

\[ \mathcal{R}(F) \leq \sqrt{\pi/2rK} \left( \sum_{l=1}^{n} \frac{1}{(w_{b(x)}^2)^{1/2}} + \frac{2\sqrt{n}}{\min_{x \in X} w_{b(x)}} + \sum_{l=1}^{n} (w_{b(x)}^2)^{1/2} \frac{1}{\min_{x \in X} (w_{b(x)}^2)} \right). \]

Proof. Let \( M \in \mathbb{R}^{\sum_{i=1}^{r} K_i \times K} \), whose \( k \)-th column represents the \( k \)-th centroid \( m_k \). Define the Gaussian processes indexed by \( M \) as

\[
\Omega_{M} = \sum_{l=1}^{n} \gamma_{l} \min_{k} w_{b(x)} \left\| \frac{b(x_l)}{w_{b(x)}} - M_{e_k} \right\|^2
\]

\[
\Xi_{M} = \sum_{l=1}^{n} \sum_{k=1}^{K} \gamma_{lk} w_{b(x)} \left( \frac{b(x_l)}{w_{b(x)}} - M_{e_k} \right)^2,
\]

where \( \gamma_{l} \) and \( \gamma_{lk} \) are independent Gaussian random variables indexed by \( l \) and \( k \). And \( e_k \) are the natural bases indexed by \( k \).

For any \( M \) and \( M' \), we have

\[
E(\Omega_{M} - \Omega_{M'})^2
\]

\[
= \sum_{l=1}^{n} \left( \min_{k} w_{b(x)} \right) \left\| \frac{b(x_l)}{w_{b(x)}} - M_{e_k} \right\|^2 \left( \min_{k} w_{b(x)} \right) \left\| \frac{b(x_l)}{w_{b(x)}} - M'_{e_k} \right\|^2
\]

\[
\leq \sum_{l=1}^{n} \left( \max_{k} w_{b(x)} \right) \left\| \frac{b(x_l)}{w_{b(x)}} - M_{e_k} \right\|^2 \left( \max_{k} w_{b(x)} \right) \left\| \frac{b(x_l)}{w_{b(x)}} - M'_{e_k} \right\|^2
\]

\[
\leq \sum_{l=1}^{n} \sum_{k=1}^{K} \left( w_{b(x)} \right) \left( \frac{b(x_l)}{w_{b(x)}} - M_{e_k} \right)^2 - w_{b(x)} \left( \frac{b(x_l)}{w_{b(x)}} - M'_{e_k} \right)^2
\]

\[
= E(\Xi_{M} - \Xi_{M'})^2.
\]

Note that the first and last inequalities hold because of the orthogaussian properties.
Using Slepian’s Lemma and Lemma A.9.2 we have

\[ \mathcal{R}(F_{\Pi_K}) \]
\[ = E_\sigma \sup_{M} \frac{1}{n} \sum_{l=1}^{n} \sigma_l \min_k w_{b(x_l)} \left\| \frac{b(x_l)}{w_{b(x_l)}} - M e_k \right\|^2 \]
\[ \leq E_\gamma \frac{\sqrt{\pi/2}}{n} \sup_{M} \sum_{l=1}^{n} \gamma_l \min_k w_{b(x_l)} \left\| \frac{b(x_l)}{w_{b(x_l)}} - M e_k \right\|^2 \]
\[ = \frac{\sqrt{\pi/2}}{n} E_\gamma \left( \sup_{M} \sum_{l=1}^{n} \sum_{k=1}^{K} \gamma_{lk} w_{b(x_l)} \left( \left\| \frac{b(x_l)}{w_{b(x_l)}} \right\|^2 - 2 \left\langle \frac{b(x_l)}{w_{b(x_l)}}, M e_k \right\rangle + \left\| M e_k \right\|^2 \right) \right) \]
\[ \leq \frac{\sqrt{\pi/2}}{n} (E_\gamma \sup_{M} \sum_{l=1}^{n} \sum_{k=1}^{K} \gamma_{lk} r \frac{1}{w_{b(x_l)}} + 2E_\gamma \sup_{M} \sum_{l=1}^{n} \sum_{k=1}^{K} \gamma_{lk} \left\langle b(x_l), M e_k \right\rangle + E_\gamma \sup_{M} \sum_{l=1}^{n} \sum_{k=1}^{K} \gamma_{lk} w_{b(x_l)} \left\| M e_k \right\|^2). \]

We give upper bounds to the three terms respectively.

\[ E_\gamma \sup_{M} \sum_{l=1}^{n} \sum_{k=1}^{K} \gamma_{lk} r \frac{1}{w_{b(x_l)}} \]
\[ = E_\gamma r \sum_{k=1}^{K} \sum_{l=1}^{n} \gamma_{lk} \frac{1}{w_{b(x_l)}} = E_\gamma r \sum_{k=1}^{K} \sqrt{\left( \sum_{l=1}^{n} \frac{\gamma_{lk}}{w_{b(x_l)}} \right)^2} \]
\[ \leq r \sum_{k=1}^{K} \sqrt{\frac{\sum_{l=1}^{n} \frac{1}{(w_{b(x_l)})^2}}{\sum_{l=1}^{n} \frac{1}{(w_{b(x_l)})^2}}} = r K \sqrt{\frac{\sum_{l=1}^{n} \frac{1}{(w_{b(x_l)})^2}}{\sum_{l=1}^{n} \frac{1}{(w_{b(x_l)})^2}}}. \]

Note that the last inequality holds for the Jensen’s inequality and the orthogaussian property of the Gaussian random variable. We therefore have

\[ 2E_\gamma \sup_{M} \sum_{l=1}^{n} \sum_{k=1}^{K} \gamma_{lk} \left\langle b(x_l), M e_k \right\rangle \]
\[ \leq 2E_\gamma K \sum_{k=1}^{K} \left\| \sum_{l=1}^{n} \gamma_{lk} b(x_l) \right\| \frac{\sqrt{n}}{\min_{x \in X} w_{b(x)}} \]
\[ \leq 2 \sum_{k=1}^{K} \left( \sum_{l=1}^{n} \left\| b(x_l) \right\|^2 \right)^{\frac{1}{2}} \frac{\sqrt{n}}{\min_{x \in X} w_{b(x)}} \]
\[ = \frac{2 \sqrt{n} r K}{\min_{x \in X} w_{b(x)}}. \]
APPENDIX A. APPENDIX

The second inequality holds because
\[ \|M e_k\| = \|\sum_{b(x) \in C_k} b(x)\| \leq \sum_{b(x) \in C_k} \|b(x)\| \leq \max_{x} \|b(x)\| \sqrt{n} \]
\[ \leq \frac{\min_{x \in X} w_b(x)}{\sqrt{n}}. \]

For the upper bound \( E_{\gamma} \sup_{M} \sum_{l=1}^{n} \sum_{k=1}^{K} \gamma_{lk} w_{b(x)} \|M e_k\|^2 \),
\[ E_{\gamma} \sup_{M} \sum_{l=1}^{n} \sum_{k=1}^{K} \gamma_{lk} w_{b(x)} \|M e_k\|^2 \]
\[ \leq E_{\gamma} \sum_{k=1}^{K} \sum_{l=1}^{n} \gamma_{lk} w_{b(x)} (\frac{\sqrt{n}}{\min_{x \in X} w_b(x)})^2 \]
\[ \leq \sum_{k=1}^{K} \sum_{l=1}^{n} (w_b(x))^{\frac{1}{2}} (\frac{\sqrt{n}}{\min_{x \in X} w_b(x)})^2 \]
\[ = rK (\sum_{l=1}^{n} (w_b(x))^{\frac{1}{2}} \frac{1}{\min_{x \in X} w_b(x)^2}). \]

Thus, we have
\[ R(F_{\Pi_K}) \]
\[ \leq \frac{\sqrt{\pi/2}}{n} (rK (\sum_{l=1}^{n} (w_b(x))^{\frac{1}{2}} + \frac{2\sqrt{n}rK}{\min_{x \in X} w_b(x)} + rK (\sum_{l=1}^{n} (w_b(x))^{\frac{1}{2}} \frac{1}{\min_{x \in X} w_b(x)} ) \]
\[ = \frac{\sqrt{\pi/2}}{n} (rK (\sum_{l=1}^{n} (w_b(x))^{\frac{1}{2}} + \frac{2\sqrt{n}rK}{\min_{x \in X} w_b(x)} + (\sum_{l=1}^{n} (w_b(x))^{\frac{1}{2}} \frac{1}{\min_{x \in X} w_b(x)} ) \]
This concludes the proof of Lemma A.9.3.

Theorem 4 in the paper thus follows according to Theorem A.3 and Lemma 4.

A.10 Proof of Theorems 3.2.3 and 3.3.3

Proof. It has been proven that the co-associate matrix \( S \) will converge \( w.r.t. \) \( r \), the number of basic crisp partitions [82]. That is, for any \( \lambda_1 > 0 \), there exists a matrix \( S_0 \), such that
\[ \lim_{r \to \infty} \Pr\{\|S - S_0\| \geq \lambda_1\} \to 0. \]

Thus, according to the definition of \( b(x) \) and Theorem 3.1.1, we can claim that \( b(x) \) and the centroids \( m_1, \ldots, m_K \) will converge to some \( b_0(x) \) and \( m^0_1, \ldots, m^0_K \), respectively, as \( r \) goes to infinity. Then,
for any $\lambda > 0$, there exist a clustering $\pi_0$ such that

$$\lim_{r \to \infty} \Pr\{|\pi - \pi_0| \geq \lambda\} \to 0,$$

which concludes the proof of Theorem 3.2.3.

Since the proof of the convergence property of the co-associate matrix $S$ also holds for the incomplete basic partitions, Theorem 3.3.3 can be easily proven by the same proof method of Theorem 3.2.3.

A.11 Proof of Theorem 3.3.1

Proof. The proof of Theorem 3.3.1 is similar to the proof of Theorem 3.1.2, with the only difference being that the missing elements are not taken into account in the objective function of weighted K-means clustering. We therefore have:

$$\sum_{x \in X} f_{m_1,...,m_K}(x) = \sum_{i=1}^{r} \sum_{k=1}^{K} \sum_{x \in C_k \cap X_i} w_{b(x)} \frac{b(x)_i}{w_b(x)} - m_{k,i}||^2$$

$$= \sum_{i=1}^{r} \sum_{k=1}^{K} \sum_{x \in C_k \cap X_i} \frac{||b(x)_i||^2}{w_b(x)} - 2b(x)_i m_{k,i} - w_{b(x)} ||m_{k,i}||^2$$

$$= \sum_{i=1}^{r} \sum_{k=1}^{K} \sum_{x \in C_k \cap X_i} \frac{||b(x)_i||^2}{w_b(x)} - \sum_{i=1}^{r} \sum_{k=1}^{K} w^{(i)} C_k \frac{||m_{k,i}||^2}{w_b(x)}$$

$$= \sum_{i=1}^{r} \sum_{k=1}^{K} \sum_{x \in C_k \cap X_i} \frac{||b(x)_i||^2}{w_b(x)} - n \sum_{i=1}^{r} \sum_{k=1}^{K} w^{(i)} C_k \frac{\sum_{j=1}^{K_i} \sum_{k=1}^{K} \sum_{x \in C_k \cap X_i} p_{kj}^2}{p_{k+}}.$$

According to the definition of centroids of K-means, we have $m_k = \langle m_{k,1}, \cdots, m_{k,r} \rangle$, $m_{k,i} = \sum_{x \in C_k \cap X_i} b(x)_i / \sum_{x \in C_k \cap X_i} w_b(x)$, $p^{(i)} = |X_i| / |X| = n^{(i)} / n$, $n^{(i)} = |C_k \cap X_i|$, $w^{(i)} C_k = \sum_{x \in C_k \cap X_i} w_b(x)$. By noting that $(\gamma)$ is a constant, we get the utility function of SEC with incomplete basic partitions and complete the proof.

A.12 Proof of Theorem 3.3.2

Proof. The weighted K-means iterates the assigning and updating phase. In the assigning phase, each instance is assigned to the nearest centroid and so the objective function decreases. Thus, we
APPENDIX A. APPENDIX

analyze the change of objective function during updating phase under the circumstance of SEC with incomplete basic partitions. For any centroid \( g = \langle g_1, \cdots, g_k \rangle, g_k = \langle g_{k,i}, \cdots, g_{k,r} \rangle, \) and \( g_k \neq m_k, \)

\[
\Delta = \sum_{i=1}^{r} \sum_{k=1}^{K} \sum_{x \in C_k \cap X_i} w_{b(x)} \left[ \| b(x)_i - g_{k,i} \|^2 - \| b(x)_i - m_{k,i} \|^2 \right]. \tag{A.20}
\]

According to the Bergman divergence \[194\], \( f(a, b) = \| a - b \|^2 = \phi(a) - \phi(b) - (a - b)^T \nabla \phi(b), \) where \( \phi(a) = \| a \|^2, \) Eq. A.20 can be rewritten as follows:

\[
\Delta = \sum_{i=1}^{r} \sum_{k=1}^{K} \sum_{x \in C_k \cap X_i} w_{b(x)} \left[ \phi(b(x)_i) - \phi(g_{k,i}) + (b(x)_i - g_{k,i})^T \nabla(g_{k,i}) - \phi(b(x)_i) + \phi(m_{k,i}) - (b(x)_i - m_{k,i})^T \nabla(m_{k,i}) \right]
\]

\[
= \sum_{i=1}^{r} \sum_{k=1}^{K} \sum_{x \in C_k \cap X_i} w_{b(x)} \left[ \phi(m_{k,i}) - \phi(g_{k,i}) + (b(x)_i - g_{k,i})^T \nabla(g_{k,i}) \right]
\]

\[
= \sum_{i=1}^{r} \sum_{k=1}^{K} w_{C_k} (1) \| m_{k,i} - g_{k,i} \|^2 > 0.
\]

Hence, the objective value will decrease during the update phase as well. Given the finite solution space, the iteration will converge within finite steps. We complete the proof. ■

A.13 Proof of Theorem 5.3.1

Proof. We start from the objective function of K-means.

\[
\sum_{k=1}^{K} \sum_{d_i \in C_k} f(d_i, m_k)
\]

\[
= \sum_{k=1}^{K} \sum_{d_i \in C_k \cap S} (\| d_i^{(1)} - m_k^{(1)} \|^2 + \lambda \| d_i^{(2)} - m_k^{(2)} \|^2) + \sum_{k=1}^{K} \sum_{d_i \in C_k \cap S} \| d_i^{(1)} - m_k^{(1)} \|^2 \tag{A.21}
\]

\[
= \| X_1 - H_1 C \|_F^2 + \lambda \| S - H_1 G \|_F^2 + \| X_2 - H_2 C \|_F^2.
\]

According to the definition of the augmented matrix \( D, \) we finish the proof.
APPENDIX A. APPENDIX

A.14 Proof of Theorem 6.2.1

We start from the objective function of K-means.

\[
\sum_{k=1}^{K} \sum_{d_i \in C_k} f(d_i, m_k) = \sum_{k=1}^{K} \left( \sum_{d_i \in C_k \cap Z_1} ||d_i^{(1)} - m_k^{(1)}||_2^2 + \sum_{d_i \in C_k \cap Y_{S1}} ||d_i^{(2)} - m_k^{(2)}||_2^2 \right) \\
+ \sum_{d_i \in C_k \cap Z_2} ||d_i^{(3)} - m_k^{(3)}||_2^2 + \sum_{d_i \in C_k \cap Y_{S2}} ||d_i^{(4)} - m_k^{(4)}||_2^2 \\
= ||Z_{S1} - H_{S1} G_1||_F^2 + ||Z_{T1} - H_{T} G_1||_F^2 + \lambda ||Y_{S1} - H_{S1} M_1||_F^2 \\
+ ||Z_{S2} - H_{S2} G_2||_F^2 + ||Z_{T2} - H_{T} G_2||_F^2 + \lambda ||Y_{S2} - H_{S2} M_2||_F^2 \\
\tag{A.22}
\]

According to the definition of \(D, Z_{S1}, Z_{S2}, Z_{T1}, Z_{T2}, H_{S1}, H_{S2}, H_{T}\) and Eq. (6.10), we finish the proof.

A.15 Survival analysis of IEC

Table A.1: Survival analysis of different clustering algorithms on protein expression data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>AL</th>
<th>SL</th>
<th>CL</th>
<th>KM</th>
<th>SC</th>
<th>LCE</th>
<th>ASRS</th>
<th>IEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLCA</td>
<td>0.8400</td>
<td>0.6230</td>
<td>0.3210</td>
<td>0.0241</td>
<td>0.0005</td>
<td>0.0881</td>
<td>0.1030</td>
<td>0.0008</td>
</tr>
<tr>
<td>BRCA</td>
<td>0.2660</td>
<td>0.0008</td>
<td>0.0988</td>
<td>0.0997</td>
<td>0.1130</td>
<td>0.3060</td>
<td>0.1460</td>
<td>0.0092</td>
</tr>
<tr>
<td>COAD</td>
<td>0.8750</td>
<td>0.0005</td>
<td>0.0843</td>
<td>0.0157</td>
<td>0.0738</td>
<td>1.20E-8</td>
<td>4.82E-5</td>
<td>1.50E-8</td>
</tr>
<tr>
<td>HNSC</td>
<td>0.7540</td>
<td>0.0050</td>
<td>0.5520</td>
<td>0.7340</td>
<td>0.5110</td>
<td>0.9840</td>
<td>0.5960</td>
<td>0.1340</td>
</tr>
<tr>
<td>KIRC</td>
<td>0.7640</td>
<td>0.9140</td>
<td>0.2460</td>
<td>0.4120</td>
<td>0.6560</td>
<td>0.1680</td>
<td>0.7590</td>
<td>0.0003</td>
</tr>
<tr>
<td>LGG</td>
<td>0.0182</td>
<td>0.0305</td>
<td>0.0002</td>
<td>0.0563</td>
<td>0.0198</td>
<td>0.0094</td>
<td>0.1780</td>
<td>0.0004</td>
</tr>
<tr>
<td>LUAD</td>
<td>0.3730</td>
<td>0.0008</td>
<td>0.3200</td>
<td>0.4790</td>
<td>0.3990</td>
<td>0.0293</td>
<td>0.5070</td>
<td>0.0267</td>
</tr>
<tr>
<td>LUSC</td>
<td>0.9050</td>
<td>0.0929</td>
<td>0.9340</td>
<td>0.6670</td>
<td>0.6050</td>
<td>0.6550</td>
<td>0.5420</td>
<td>0.0982</td>
</tr>
<tr>
<td>OV</td>
<td>0.8090</td>
<td>0.5450</td>
<td>0.1900</td>
<td>0.0275</td>
<td>0.0446</td>
<td>0.0485</td>
<td>0.0327</td>
<td>0.0026</td>
</tr>
<tr>
<td>PRAD</td>
<td>1.19E-6</td>
<td>9.75E-7</td>
<td>3.16E-6</td>
<td>0.0011</td>
<td>0.0918</td>
<td>0.8140</td>
<td>0.0124</td>
<td>0.0041</td>
</tr>
<tr>
<td>SKCM</td>
<td>0.0848</td>
<td>0.2860</td>
<td>0.0100</td>
<td>0.0929</td>
<td>0.0411</td>
<td>0.0381</td>
<td>0.0059</td>
<td>3.00E-4</td>
</tr>
<tr>
<td>THCA</td>
<td>0.2380</td>
<td>0.0255</td>
<td>0.3470</td>
<td>0.1910</td>
<td>0.1480</td>
<td>0.0799</td>
<td>0.1370</td>
<td>0.0187</td>
</tr>
<tr>
<td>UCEC</td>
<td>0.4530</td>
<td>3.00E-8</td>
<td>0.9860</td>
<td>0.9860</td>
<td>0.4550</td>
<td>0.8450</td>
<td>0.3700</td>
<td>0.2930</td>
</tr>
</tbody>
</table>

#Significance | 2 | 6 | 3 | 4 | 3 | 5 | 4 | 10

Note: the values in the table represent the p-value of log-rank test.
### Table A.2: Survival analysis of different clustering algorithms on miRNA expression data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>AL</th>
<th>SL</th>
<th>CL</th>
<th>KM</th>
<th>SC</th>
<th>LCE</th>
<th>ASRS</th>
<th>IEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLCA</td>
<td>0.2780</td>
<td>0.5880</td>
<td>0.5940</td>
<td>0.0616</td>
<td>0.5620</td>
<td>0.3410</td>
<td>0.2400</td>
<td>0.0490</td>
</tr>
<tr>
<td>BRCA</td>
<td>0.3110</td>
<td>0.6350</td>
<td>0.5410</td>
<td>1.53E-5</td>
<td>0.0717</td>
<td>3.97E-6</td>
<td>1.12E-7</td>
<td>5.15E-7</td>
</tr>
<tr>
<td>COAD</td>
<td>0.3290</td>
<td>0.6430</td>
<td>0.2070</td>
<td>0.2290</td>
<td>0.1960</td>
<td>8.88E-4</td>
<td>0.0246</td>
<td>0.0002</td>
</tr>
<tr>
<td>HNSC</td>
<td>0.8900</td>
<td>0.8820</td>
<td>0.7650</td>
<td>0.5760</td>
<td>0.6770</td>
<td>0.0605</td>
<td>4.45E-5</td>
<td>0.0048</td>
</tr>
<tr>
<td>KIRC</td>
<td>0.7970</td>
<td>0.6420</td>
<td>0.0692</td>
<td>0.0120</td>
<td>0.0717</td>
<td>3.97E-6</td>
<td>1.12E-7</td>
<td>0.0140</td>
</tr>
<tr>
<td>LGG</td>
<td>0.8820</td>
<td>0.9640</td>
<td>0.8940</td>
<td>0.9850</td>
<td>0.9000</td>
<td>0.7450</td>
<td>0.0460</td>
<td>0.0050</td>
</tr>
<tr>
<td>LUAD</td>
<td>0.8350</td>
<td>0.1200</td>
<td>0.7410</td>
<td>0.2870</td>
<td>0.3580</td>
<td>0.0038</td>
<td>0.8260</td>
<td>0.0140</td>
</tr>
<tr>
<td>LUSC</td>
<td>0.1060</td>
<td>0.3450</td>
<td>0.0565</td>
<td>0.0034</td>
<td>0.0089</td>
<td>0.1310</td>
<td>0.3120</td>
<td>0.0002</td>
</tr>
<tr>
<td>OV</td>
<td>0.5540</td>
<td>0.0007</td>
<td>0.2410</td>
<td>0.6290</td>
<td>0.4190</td>
<td>0.2340</td>
<td>0.2300</td>
<td>0.0025</td>
</tr>
<tr>
<td>PRAD</td>
<td>0.4570</td>
<td>0.4250</td>
<td>0.6500</td>
<td>0.3300</td>
<td>0.3200</td>
<td>0.8720</td>
<td>0.6270</td>
<td>0.0190</td>
</tr>
<tr>
<td>SKCM</td>
<td>0.0619</td>
<td>0.8670</td>
<td>0.4920</td>
<td>0.6390</td>
<td>0.6940</td>
<td>0.0663</td>
<td>0.0575</td>
<td>0.0440</td>
</tr>
<tr>
<td>THCA</td>
<td>0.4660</td>
<td>0.0064</td>
<td>0.0053</td>
<td>0.0892</td>
<td>0.1100</td>
<td>0.0119</td>
<td>0.0157</td>
<td>2.95E-5</td>
</tr>
<tr>
<td>UCEC</td>
<td>0.5280</td>
<td>0.4570</td>
<td>0.6290</td>
<td>0.6870</td>
<td>0.6080</td>
<td>0.5530</td>
<td>0.3520</td>
<td>0.0258</td>
</tr>
</tbody>
</table>

#Significance: 0 2 1 2 2 5 4 11

Note: the values in the table represent the p-value of log-rank test.

### Table A.3: Survival analysis of different clustering algorithms on mRNA expression data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>AL</th>
<th>SL</th>
<th>CL</th>
<th>KM</th>
<th>SC</th>
<th>LCE</th>
<th>ASRS</th>
<th>IEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLCA</td>
<td>1.06E-7</td>
<td>8.88E-8</td>
<td>1.06E-7</td>
<td>0.0258</td>
<td>0.6860</td>
<td>0.1280</td>
<td>0.0938</td>
<td>5.53E-6</td>
</tr>
<tr>
<td>BRCA</td>
<td>5.35E-3</td>
<td>0.1740</td>
<td>0.0401</td>
<td>0.1760</td>
<td>0.0840</td>
<td>0.5980</td>
<td>0.0155</td>
<td>0.0002</td>
</tr>
<tr>
<td>COAD</td>
<td>0.8930</td>
<td>0.8960</td>
<td>0.8720</td>
<td>0.0163</td>
<td>0.0296</td>
<td>0.0048</td>
<td>0.0743</td>
<td>0.0028</td>
</tr>
<tr>
<td>HNSC</td>
<td>0.2950</td>
<td>8.53E-5</td>
<td>0.1350</td>
<td>0.7470</td>
<td>0.5440</td>
<td>0.6290</td>
<td>0.1440</td>
<td>0.0392</td>
</tr>
<tr>
<td>KIRC</td>
<td>0.0025</td>
<td>0.0012</td>
<td>0.0036</td>
<td>0.0612</td>
<td>0.1450</td>
<td>0.2420</td>
<td>0.1550</td>
<td>0.0038</td>
</tr>
<tr>
<td>LGG</td>
<td>0.0156</td>
<td>0.0156</td>
<td>0.0155</td>
<td>0.1270</td>
<td>0.1230</td>
<td>0.2650</td>
<td>0.0023</td>
<td>0.0055</td>
</tr>
<tr>
<td>LUAD</td>
<td>0.0109</td>
<td>0.8290</td>
<td>0.3190</td>
<td>0.0429</td>
<td>0.0034</td>
<td>0.0189</td>
<td>0.0157</td>
<td>0.0165</td>
</tr>
<tr>
<td>LUSC</td>
<td>0.0990</td>
<td>0.2100</td>
<td>0.0241</td>
<td>0.0355</td>
<td>0.4740</td>
<td>0.0769</td>
<td>0.1360</td>
<td>0.0371</td>
</tr>
<tr>
<td>OV</td>
<td>0.2210</td>
<td>4.92E-10</td>
<td>0.1700</td>
<td>0.6360</td>
<td>0.3780</td>
<td>0.8720</td>
<td>0.7660</td>
<td>0.4530</td>
</tr>
<tr>
<td>PRAD</td>
<td>4.29E-9</td>
<td>4.49E-9</td>
<td>5.88E-9</td>
<td>7.29E-11</td>
<td>4.10E-9</td>
<td>0.0070</td>
<td>6.75E-13</td>
<td>0.0001</td>
</tr>
<tr>
<td>SKCM</td>
<td>0.0012</td>
<td>0.0012</td>
<td>0.0015</td>
<td>0.5230</td>
<td>0.0006</td>
<td>0.1350</td>
<td>5.91E-10</td>
<td>0.0204</td>
</tr>
<tr>
<td>THCA</td>
<td>0.0147</td>
<td>0.5650</td>
<td>0.0713</td>
<td>0.0244</td>
<td>0.0561</td>
<td>0.2380</td>
<td>0.0710</td>
<td>0.0048</td>
</tr>
<tr>
<td>UCEC</td>
<td>0.5790</td>
<td>0.0594</td>
<td>0.1930</td>
<td>0.1850</td>
<td>0.2460</td>
<td>0.3670</td>
<td>0.4890</td>
<td>0.0437</td>
</tr>
</tbody>
</table>

#Significance: 8 7 7 6 4 3 5 10

Note: the values in the table represent the p-value of log-rank test.
Table A.4: Survival analysis of different clustering algorithms on SCNA data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>AL</th>
<th>SL</th>
<th>CL</th>
<th>KM</th>
<th>SC</th>
<th>LCE</th>
<th>ASRS</th>
<th>IEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLCA</td>
<td>0.3710</td>
<td>0.3710</td>
<td>0.3810</td>
<td>0.6340</td>
<td>0.3580</td>
<td>0.4340</td>
<td>0.3800</td>
<td>0.0120</td>
</tr>
<tr>
<td>BRCA</td>
<td>0.6540</td>
<td>0.6540</td>
<td>0.1160</td>
<td>0.0090</td>
<td>0.4790</td>
<td>0.0798</td>
<td>0.3520</td>
<td>0.0073</td>
</tr>
<tr>
<td>COAD</td>
<td>0.9320</td>
<td>0.9320</td>
<td>0.9010</td>
<td>0.1600</td>
<td>0.7920</td>
<td>0.7670</td>
<td>0.4660</td>
<td>0.3900</td>
</tr>
<tr>
<td>HNSC</td>
<td>0.0003</td>
<td>0.0003</td>
<td>0.0380</td>
<td>0.5280</td>
<td>0.5730</td>
<td>0.8280</td>
<td>0.7710</td>
<td>0.2940</td>
</tr>
<tr>
<td>KIRC</td>
<td>0.6580</td>
<td>0.7510</td>
<td>0.0929</td>
<td>0.4390</td>
<td>0.1060</td>
<td>0.2690</td>
<td>0.3710</td>
<td>0.2210</td>
</tr>
<tr>
<td>LGG</td>
<td>0.8800</td>
<td>0.9950</td>
<td>0.6430</td>
<td>0.5710</td>
<td>0.6130</td>
<td>0.8750</td>
<td>0.9740</td>
<td>0.4930</td>
</tr>
<tr>
<td>LUAD</td>
<td>0.5420</td>
<td>0.5420</td>
<td>0.5880</td>
<td>0.0763</td>
<td>0.2390</td>
<td>0.0121</td>
<td>0.0080</td>
<td>0.0456</td>
</tr>
<tr>
<td>LUSC</td>
<td>0.9890</td>
<td>0.8190</td>
<td>0.3870</td>
<td>0.3560</td>
<td>0.3810</td>
<td>0.1710</td>
<td>0.5540</td>
<td>0.1290</td>
</tr>
<tr>
<td>OV</td>
<td>0.7500</td>
<td>0.7500</td>
<td>0.1270</td>
<td>0.1710</td>
<td>0.0904</td>
<td>0.1730</td>
<td>0.1380</td>
<td>1.08E-7</td>
</tr>
<tr>
<td>PRAD</td>
<td>0.8410</td>
<td>2.40E-7</td>
<td>0.5060</td>
<td>0.2640</td>
<td>0.0008</td>
<td>0.0160</td>
<td>0.0046</td>
<td>0.0003</td>
</tr>
<tr>
<td>SKCM</td>
<td>0.8730</td>
<td>0.8140</td>
<td>0.6790</td>
<td>0.5660</td>
<td>0.1970</td>
<td>0.2210</td>
<td>0.2040</td>
<td>0.0444</td>
</tr>
<tr>
<td>THCA</td>
<td>0.1530</td>
<td>0.5180</td>
<td>0.1440</td>
<td>0.2670</td>
<td>0.1960</td>
<td>0.1360</td>
<td>0.5440</td>
<td>0.0496</td>
</tr>
<tr>
<td>UCEC</td>
<td>0.1100</td>
<td>0.1100</td>
<td>0.2310</td>
<td>0.0484</td>
<td>0.0673</td>
<td>0.4860</td>
<td>0.3450</td>
<td>0.1210</td>
</tr>
</tbody>
</table>

#Significance | 1 | 2 | 1 | 2 | 1 | 2 | 2 | 7

Note: the values in the table represent the p-value of log-rank test.

Table A.5: Survival analysis of IEC on pan-omics gene expression.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>BLCA</th>
<th>LGG</th>
<th>SKCM</th>
<th>BLCA</th>
<th>COAD</th>
<th>HNSC</th>
<th>KIRC</th>
<th>PRAD</th>
<th>THCA</th>
<th>UCEC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.0041</td>
<td>0.0054</td>
<td>4.14E-4</td>
<td>0.0327</td>
<td>0.0160</td>
<td>0.0178</td>
<td>1.92E-8</td>
<td>0.0423</td>
<td>0.0054</td>
<td>1.58E-4</td>
</tr>
</tbody>
</table>

Note: the values in the table represent the p-value of log-rank test.