AN ABSTRACT OF THE DISSERTATION OF

Yuanli Pei for the degree of Doctor of Philosophy in Computer Science presented on June 16, 2017.

Title: Learning with Partial Supervision for Clustering and Classification

Abstract approved: _____

Xiaoli Z. Fern

In the field of machine learning, clustering and classification are two fundamental tasks. Traditionally, clustering is an unsupervised method, where no supervision about the data is available for learning; classification is a supervised task, where fully-labeled data are collected for training a classifier. In some scenarios, however, we may not have the full label but only partial supervision about the data, such as instance similarities or incomplete label assignments. In such cases, traditional clustering and classification methods do not directly apply. To address such problems, this thesis focuses on the task of learning from partial supervision for clustering and classification tasks. For clustering with partial supervision, we investigate three problems: a) constrained clustering in multi-instance multi-label learning, where the goal is to group instances into clusters that respect the background knowledge given by the bag-level labels; b) clustering with constraints, where the partial supervision is expressed as "pairwise constraints" or "relative constraints", regarding similarities about instance pairs and triplets respectively; c) active learning of pairwise constraints for clustering, where the goal is to improve the clustering with minimum human effort by iteratively querying the most informative pairs to an oracle. For classification with partial supervision, we address the problem of multi-label learning where data is associated with a latent label hierarchy and incomplete label assignments, and the goal is to simultaneously discover the latent hierarchy as well as to learn a multi-label classifier that is consistent with the hierarchy.

Learning with Partial Supervision for Clustering and Classification

by

Yuanli Pei

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented June 16, 2017
Commencement June 2017

Doctor of Philosophy dissertation of Yuanli Pei presented on June 16, 2017.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Yuanli Pei, Author

ACKNOWLEDGEMENTS

First, I would like to express my deep and sincere gratitude to my advisor, Xiaoli Z. Fern, for her guidance, encouragement, and continuous support during my Ph.D. study. She has provided both excellent technical contributions and spiritual support to this dissertation. More importantly, she is a role model for me to learn and to follow from every aspects of my work and life.

I would like to thank Raviv Raich for providing useful discussions in our group meetings and serving on my committee. I thank the rest of my committee members: Prasad Tadepalli, Sinisa Todorovic and Byron Marshall. Thank you for being patient and helpful during the exams and meetings.

I thank all the collaborators during my study and internship. I thank Romer Rosales, Sicheng Xiong and Teresa Vania Tjahja for contributing to the project on clustering with constraints. I thank Ween-Keen Wong, Alex Groce, Arpit Christi for collaborating on another project outside of this thesis. I thank Moises Goldszmidt, Alexandros Ntoulas, Amy Tan for guiding me through the internship.

I thank the current and former members in the Bio-acoustic group: Forrest Briggs, Anh The Pham, Zeyu You, Thi Tam Nguyen, Xingyi Li, Thi Kim Phung Lai, Gaole Jin, and Qi Lou. Thank you for discussing interesting ideas during our group meetings.

I am thankful to the great colleagues and excellent faculty and staff members from the EECS department. I have learned a lot from the discussions with the colleagues and the faculty members. I thank all the staff members in the department for responding to all my requests and tracking my progress.

I acknowledge the support of the US National Science Foundation under Grant No. IIS-1055113.

Lastly, I would like to express my deepest thanks to my family for their unconditional love and endless support. I am extremely grateful to my parents for their unconditional love and their support for my education. I would not have made it this far without them. I thank my brother for being understanding and providing endless support. I thank my mother-in-law for providing the care when we were in need. I especially thank my husband, Li-Ping Liu, for his love, support and unfailing advice. Finally, I thank my son whose presence has brought to my life endless joy.

CONTRIBUTION OF AUTHORS

Thanks to my collaborators, Rómer Rosales and Teresa V. Tjahja, who have contributed to Chapter 3, Li-Ping Liu, who has contributed to Chapter 4, and Raviv Raich, who has contributed to Chapter 5.

TABLE OF CONTENTS

Page

1 Introduction 1

1.1 Introduction 1

2 Constrained Instance Clustering in Multi-Instance Multi-Label Learning 4

2.1 Introduction 4

2.2 Problem Statement 6

2.3 Bag Constraints for MIML Instance Clustering 6

2.4 Incorporate Bag Constraints to Spectral Clustering 7

2.4.1 Preliminaries on Spectral Clustering 7

2.4.2 Spectral Clustering with Bag Constraints 8

2.4.3 Relation to ML/CL Pairwise Constraints 11

2.5 Empirical Evaluation 13

2.5.1 Datasets Description 13

2.5.2 Baseline Methods 14

2.5.3 Parameter Selection 15

2.5.4 Experiments and Discussions 15

2.6 Conclusion 20

3 Comparing Clustering with Pairwise and Relative Constraints: A Unified Framework 23

3.1 Introduction 23

3.2 A Unified Framework for Clustering with Constraints 26

3.2.1 Problem Statement 26

3.2.2 The Probabilistic Model 27

3.2.3 Objective 29

3.2.4 Optimization 31

3.3 Performance of the Proposed Clustering Framework 35

3.3.1 Datasets 35

3.3.2 Baseline Methods and Evaluation Metric 36

3.3.3 Overall Performance 36

3.3.4 Soft Constraints vs. Hard Constraints 40

3.3.5 Computational Time 41

3.4 Comparing Informativeness of Constraints: A Mathematical Analysis . . . 42

3.5 Empirical Comparison of Constraints via A User Study 45

TABLE OF CONTENTS (Continued)

Page

3.5.1 User Study Design 46
3.5.2 Results and Discussions 48
3.6 Related Work 57
3.7 Conclusions 58

4 Bayesian Active Clustering with Pairwise Constraints 60
4.1 Introduction 60
4.2 Problem Statement 62
4.3 Bayesian Active Clustering 62
4.3.1 The Bayesian Clustering Model 62
4.3.2 Active Query Selection 65
4.3.3 The Sequential MCMC Sampling of W 67
4.3.4 Find the MAP Solution 69
4.4 Experiments 70
4.4.1 Dataset and Setup 71
4.4.2 Effectiveness of the Proposed Clustering Model 73
4.4.3 Effectiveness of the Overall Active Clustering Model 73
4.4.4 Analysis of the Acyclic Graph Restriction 76
4.5 Related Work 76
4.6 Conclusion 78

5 Learning with Latent Label Hierarchy from Incomplete Multi-Label Data 80
5.1 Introduction 80
5.2 Problem Statement 82
5.3 The Probabilistic Graphical Model 83
5.3.1 Model Specification 84
5.3.2 Maximum-Likelihood Objective 85
5.4 Optimization with Expectation-Maximization 85
5.4.1 E-Step 86
5.4.2 M-Step 89
5.5 Experiments 90
5.5.1 Learning Latent Label Hierarchy 90
5.5.2 Classification with Incomplete Multi-Label Data 95
5.5.3 Case Study: Application to the Sosh Dataset 98

TABLE OF CONTENTS (Continued)

Page

5.6 Related Work 99
5.7 Conclusion 100

6 Conclusions and Future Work 102
6.1 Contributions 102
6.2 Future Work 103

Bibliography 104

Appendices 117
A Appendix for Manuscript 1 118

LIST OF FIGURES

Figure Page

2.1 Class Purity results as a function of α. SP is unconstrained spectral clustering. CSP is the proposed spectral clustering with bag constraints. . 15

2.2 Scenario of $K = C$: NMI results as a function of constraints creating from different percentage of labeled bags. Error bars are reported with mean and standard deviation. 17

2.3 Scenario of $K = C$: Class Purity results as a function of different number of constraints. Error bars are reported with mean and standard deviation. 18

2.4 Scenario of $K > C$: Class Purity results as a function of the number of clusters. Error bars are reported with mean and standard deviation. . . . 19

3.1 The unified graphical model for clustering with constraints. (a) the unified model; (b) instantiation of pairwise constraints; (c) instantiation of relative constraints. For simplicity, only the variables directly connected with a single constraint label l_j are shown. When multiple labels are considered the directed graph normally has loops and exact inference is not tractable. 28

3.2 Pairwise Constraints: Clustering performance with different number of constraints. Error bars are reported with mean and the 95% confidence intervals. 38

3.3 Relative Constraints: Clustering performance with different number of constraints. Error bars are reported with mean and the 95% confidence intervals. 39

3.4 Pairwise Constraints: Clustering performance of our method using hard vs. soft constraints. 40

3.5 Relative Constraints: Clustering performance of our method using hard vs. soft constraints. 41

3.6 Comparison of the mutual information between the pairwise and relative constraint labels and the instance cluster labels as a function of the number of clusters. 45

3.7 Screenshots of obtaining pairwise and relative constraints for the Flower dataset using the developed interface. 47

LIST OF FIGURES (Continued)

Figure Page

3.8 Clustering performance with synthetic pairwise and relative constraints with the same number of instances. Results are averaged over 20 runs. Error bars show the 95% confidence intervals. 54

3.9 Clustering performance with human-labeled pairwise and relative constraints. "Same Inst": two types of constraints involve the same number of instances. "Same Time": two types of constraints are obtained within the same amount of time. Results are averaged over 24 users. Error bars show the 95% confidence intervals. 55

3.10 The number of all labeled constraints and correctly labeled constraints used in the comparison of Figure 3.9. "R": relative constraints; "R-C": $ab\bar{c}, a\bar{b}c, \bar{a}bc$ relative constraints; "P": pairwise constraints; "-all": total number of constraints;"-Correct": correctly labeled constraints. 56

4.1 Pairwise F-Measure clustering results with increasing number of randomly selected queries. Results are averaged over 30 runs. Error bars are shown as mean and 95% confidence interval. 72

4.2 Pairwise F-Measure clustering results of different active clustering methods with increasing number of queries. Results are averaged over 30 runs. Error bars are shown as mean and 95% confidence interval. 74

5.1 A graphical model for learning with latent label hierarchy from incomplete multi-label data. 83

5.2 Ground-truth Label Hierarchy of the Pascal 2007 and 20 Newsgroup datasets. 91

5.3 RF measure of the learned tree structure with different label noise ratios. Error bars are 95% confidence interval. 91

5.4 F1-Micro and F1-Macro multi-label classification performance with different label noise ratios. 94

5.5 Estimated α value of the proposed method with different label noise ratios. 96

5.6 Subtrees of the learned label hierarchy on the Sosh data using the proposed method. 98

LIST OF TABLES

Table — Page

2.1 Relation of Bag Constraints and Pairwise Constraints for $K = C$. 12

2.2 Summary of MIML Datasets Information. Single-Label bags:number of bags that contain only a single class; Multi-Label bags: number of bags that have multiple labels; Avg. Inst.: average number of instances in each bag; Avg. Bag Label: average number of class labels in each bag. 13

3.1 Pairwise constraints: $P(l_j^p|Y_{I_j}^p)$ with $Y_{I_j}^p = [y_{a_j}, y_{b_j}]$. 29

3.2 Relative constraints: $P(l_j^r|Y_{I_j}^r)$ with $Y_{I_j}^r = [y_{a_j}, y_{b_j}, y_{c_j}]$. 29

3.3 Pairwise Constraints: the values of $\tilde{Q}(l_j^p|y_i = k, I_j^p)$, $i \in I_j^p$. For simplicity, we denote $q_{\cdot k} = q(y_{\cdot_j} = k)$. 33

3.4 Relative Constraints: the values of $\tilde{Q}(l_j^r|y_i = k, I_j^r)$, $i \in I_j^r$. For simplicity, we denote $q_{\cdot k} = q(y_{\cdot_j} = k)$ and $q_{\cdot \bar{k}} = \sum_{u \neq k} q(y_{\cdot_j} = u)$. 33

3.5 Summary of Constrained Clustering Datasets Information. 35

3.6 The average time cost (mean $\pm$ std in seconds) for labeling each constraint. 48

3.7 Survey results regarding preferences between labeling pairwise and relative constraints. 49

3.8 Birdsong v2 Data: The average confusion matrix of the human labeled constraints vs. the ground truth constraint labels. The overall accuracy for each type of constraint is shown in the corresponding caption. 51

3.9 Flower Data: The average confusion matrix of the human labeled constraints vs. the ground truth constraint labels. The overall accuracy for each type of constraint is shown in the corresponding caption. 51

4.1 Summary of Active Clustering Datasets Information 71

4.2 Number of dropped pairs (shown as Info/Uncertain) required to find the best pair that does not a create cycle at different iterations. Results are averaged over 30 runs. 77

LIST OF TABLES (Continued)

Table Page

5.1 Specification of conditional probabilities for learning with latent label hierarchy from incomplete multi-label data. 84

5.2 Summary of Multi-Label Datasets Information. 90

LIST OF ALGORITHMS

Algorithm		Page
1	Spectral Clustering with MIML Bag Constraints	11
2	Bayesian Active Clustering with Pairwise Constraints	70
3	E-step: Compute $Q(\mathbf{y}_i) = P(\mathbf{y}_i \mid \mathbf{z}_i, \mathbf{x}_i, W, \mathbf{t}, \alpha)$	88
4	EM Algorithm for Learning with Latent Label Hierarchy from Incomplete Multi-Label Data	88

Chapter 1: Introduction

1.1 Introduction

In the field of machine learning, clustering and classification are two fundamental tasks. Clustering is the task of grouping a collection of objects in a way that similar objects are grouped together. It is traditionally an unsupervised task, namely, no supervision such as label information is available. In the task of classification, a set of data samples and their labels are collected, and the goal is to learn a classifier that can be later used to predict labels for unseen data. In this way, classification is a supervised problem as the data instances are fully labeled.

In some learning scenarios, we do not have the full label about the data but we may have some partial supervision. For example, for clustering tasks, we may have some information about similarities between instances in addition to the data itself. Such information could be helpful toward improving clustering. Unfortunately, they can not be incorporated by traditional unsupervised clustering methods. For classification tasks, one example is the multi-label learning where each instance can belong to multiple classes. In such problems, we may not be able to collect all the labels for each instance to train a classifier. Instead, we may only have a subset of labels for each instance. Traditional classification methods usually assume that the instance labels are complete, so they can not deal with such problem. Such challenges in both tasks motivate us to study the problem of learning with partial supervision.

In this dissertation, we solve three problems for clustering with partial supervision, sometimes called "semi-supervised clustering" in the literature. The first problem is constrained clustering in multi-instance multi-label (MIML) learning (Chapter 2). In the MIML framework, datasets are given in the form of bags, each of which contains multiple instances and is associated with multiple labels. Given a set of labeled bags, our goal is to group instances into cluster, which correspond to the class labels or subclasses within each class. We propose to encode the bag-label knowledge into "bag constraints" and demonstrate how such constraints can be incorporated into a popular spectral clustering

algorithm.

The second problem is clustering with pairwise and relative constraints (Chapter 3). In this problem, partial supervision is represented in two forms: pairwise and relative constraints, regarding similarities about instance pairs and triplets respectively. We develop a unified framework that can effectively learn from either type of constraints. In addition, this unified model also allows us to compare the usefulness of the two types of constraints. This is essential since in practice it is often complicated to acquire constraints and usually only one type of constraints can be obtained. In such cases, it is critical to understand which type of constraints is more useful. We perform extensive comparisons between these two types of constraints in a user study.

The third problem is active learning of pairwise constraints for clustering (Chapter 4). Clustering can be improved given similarity constraints on instance pairs. However, constraints on some instances can be more informative than the others. Randomly selecting instances to label their constraints could lead to wasted labeling effort or even degrade the clustering performance. We introduce a Bayesian clustering model that actively selects the most informative pair of instances to query an oracle, and iteratively updates the model posterior based on the obtained pairwise constraints. Our results demonstrate the effectiveness of the proposed method over passive learning and other active learning methods.

Finally, we study the problem of learning with latent hierarchy from incomplete multi-label data (Chapter 5). In our motivating real world problem, events are tagged with multiple labels which form a potential unknown hierarchy. In addition, different with traditional multi-label tasks where each sample is labeled with the complete set of classes it belongs to, data in our problem may have incomplete class assignments due to the inconsistency in the labeling process. Our goal is to simultaneously discover the latent hierarchy and learn a multi-label classifier that is consistent with the hierarchy. We propose a graphical model that captures the labeling process of the data. By specifying appropriate conditional probabilities, we can capture the hierarchy structure and the label incompleteness. We propose a maximum likelihood estimation approach to efficiently estimate the model parameters and learn the hierarchy structure. Our results on real-world datasets demonstrate that our method can both learn an effective multi-label classifier as well as discover interesting label hierarchy from the incomplete data.

Manuscript 1: Constrained Instance Clustering in Multi-Instance Multi-Label Learning

Yuanli Pei and Xiaoli Z. Fern

Pattern Recognition Letters
Edited By Friedhelm Schwenker and Edmondo Trentin
Published by Elsevier
Volume 37, 2014, Pages 107–114.

Chapter 2: Constrained Instance Clustering in Multi-Instance Multi-Label Learning

Abstract

In multi-instance multi-label (MIML) learning, datasets are given in the form of bags, each of which contains multiple instances and is associated with multiple labels. This chapter considers a novel instance clustering problem in MIML learning, where the bag labels are used as background knowledge to help group instances into clusters. The goal is to recover the class labels or to find the subclasses within each class. Prior work on constraint-based clustering focuses on pairwise constraints and can not fully utilize the bag-level label information. We propose to encode the bag-label knowledge into soft bag constraints that can be easily incorporated into any optimization based clustering algorithm. As a specific example, we demonstrate how the bag constraints can be incorporated into a popular spectral clustering algorithm. Empirical results on both synthetic and real-world datasets show that the proposed method achieves promising performance compared to state-of-the-art methods that use pairwise constraints.

2.1 Introduction

The Multi-Instance Multi-Label (MIML) learning framework [129] has been successfully applied in a variety of applications including computer vision [40, 114, 124] and audio analysis [113]. In MIML, datasets are given in the form of bags and each bag contains multiple instances. It is assumed that there exists a class structure such that each instance in the bag belongs to one of the classes. However, the instance class labels are not directly observed. Instead, the class labels are only provided at the bag level, which is the union of all instance labels within the bags. The goal of MIML learning is then to build a classifier to predict the labels for an unseen bag [127, 129] or to annotate the label of each instance within the bag [16].

In this chapter, we consider a novel instance clustering problem within the MIML framework, where the goal is to group instances from all bags into clusters. In particular, we seek to find a cluster structure that corresponds to or refines the existing class

structure. That is, we assume that each class contains one or more subclasses and our goal is to find such subclasses via clustering. In our motivating application, we want to understand the structure of bird song within each species. Here a bag corresponds to the spectrogram of a 10-second field recording of multiple birds, and each instance corresponds to a segment in the spectrogram capturing a single bird utterance (a syllable). The labels of a bag are the set of species (one or more) present in the recording. Birds from a single species may vocalize in different modes. For instance, the sound made by a woodpecker has at least two distinct modes: pecking and calling. We are interested in finding such distinct modes within each species by applying clustering techniques to instances. Ideally we would perform clustering on instances of the same species to learn such modes. However, this is impractical because the labels are only provided at the bag level and we do not have accurate instance-level species labels. Therefore, we cast this problem as an instance clustering problem with bag-level class labels as side information.

Existing literature on clustering with side information primarily focuses on pairwise Must-Link (ML) and Cannot-Link (CL) constraints [51, 53, 57, 104, 106, 119, 121]. Note that one could potentially generate ML and CL constraints based on the bag-level labels, but they incorporate only limited information for MIML datasets (as will be discussed in Sec.2.4.3) and are not effective for our problem. Another closely related topic is MIML instance annotation [16, 116, 124], where an instance classifier is learned from MIML data that predicts the class label of each instance. The key difference between MIML instance annotation and our work is that we are interested in finding the refinement of class structure for the instances, whereas instance annotation only focuses on recovering the class labels of instances based-on the bag-level labels.

In this chapter, we propose to incorporate the bag-level side information in the form of *bag constraints*. Our approach defines two similarity measures between bags based on *class* labels and *cluster* labels respectively. By requiring the two similarities to order pairs of bags consistently, we encode bag-level label knowledge into soft constraints, which can be easily incorporated into traditional clustering objectives as a penalty term. In particularly, we incorporate such constraints into a popular spectral clustering algorithm and validate the effectiveness of the resulting method on both synthetic and real-world datasets. Experiments show that our method produces good clustering results compared to spectral clustering methods with pairwise constraints.

2.2 Problem Statement

In our problem, the data consists of M bags $\{\mathbb{B}_1, \cdots, \mathbb{B}_M\}$, where each bag $\mathbb{B}_i$ contains n_i instances, i.e., $\mathbb{B}_i = \{x_{i1}, \cdots, x_{in_i}\}$ with $x_{iq} \in \mathcal{R}^d$. As prior knowledge, each $\mathbb{B}_i$ is associated with a set of class labels, denoted by $\mathbb{Y}_i \subseteq \{1, \cdots, C\}$, where C is the total number of distinct classes. Denote $\mathcal{X} = \bigcup_{m=1}^{M} \mathbb{B}_m$ and let $N = \sum_{m=1}^{M} n_m$ be the total number of instances[1] in $\mathcal{X}$, our goal is to partition the N instances in $\mathcal{X}$ into K disjoint clusters that respect the class boundaries. That is, if x_p and x_q belong to the same cluster, they must belong to the same class, while the converse is true only if $K = C$, in which case we wish to recover the classes perfectly by clustering. In the case of $K > C$, some classes may contain multiple clusters that correspond to subclasses of the existing classes.

2.3 Bag Constraints for MIML Instance Clustering

In our setup, the desired cluster labels are closely related to the class labels. To capture this relationship, we introduce two different representations for each pair of bags using their class-label set and cluster-label set respectively, and require these two representations to induce similarities that behave similarly in terms of their ranking orders. That is, if a pair of bags $\mathbb{B}_i$ and $\mathbb{B}_j$ is more similar to each other than another pair $\mathbb{B}_r$ and $\mathbb{B}_s$ according to their class labels, the similarity should maintain the same order when measured using cluster labels. This will allow us to find a clustering solution that implicitly respects the class labels.

More formally, we use (i, j) to represent a pair of bags $\mathbb{B}_i$ and $\mathbb{B}_j$. Let $\Omega_L(i, j)$ be the *class-label similarity* between $\mathbb{B}_i$ and $\mathbb{B}_j$, and let $\Omega_A(i, j)$ be their *cluster-label similarity*.[2] Conceivably, a good clustering result is such that a large value of $\Omega_L(i, j)$ corresponds to a large value of $\Omega_A(i, j)$. For example, for a pair of bags $\mathbb{B}_i$ and $\mathbb{B}_j$ with a certain number of class labels, the more class labels they share, the larger the value $\Omega_L(i, j)$ is, and correspondingly we expect the value $\Omega_A(i, j)$ to be larger.

[1] In this chapter, we assume that all instances are distinct.

[2] At this point, we do not specify the function forms of $\Omega_L(\cdot, \cdot)$ and $\Omega_A(\cdot, \cdot)$, since they can be problem-specified. However, this does not prevent us from viewing them as geometrical similarities.

Using the above defined notation, we introduce the bag constraints as follows:

$$[\Omega_L(i,j) - \Omega_L(r,s)][\Omega_A(i,j) - \Omega_A(r,s)] \geq 0, \ \forall i,j,r,s \in \{1,\ldots,M\} \tag{2.1}$$

The first term on the left hand side of the above inequality compares the difference of class-label similarities between (i,j) and (r,s). The second term computes the corresponding difference of the cluster-label similarities. By requiring the nonnegativity of the product, the inequality requires the two similarities to consistently order any pairs of bags. In this way, the bag constraints indirectly enforces the consistency between class labels and cluster labels for all bags.

The above bag constraints can be easily incorporated into any optimization based clustering algorithm. Let f_A be the objective to be maximized by a clustering algorithm, the bag constraints can be incorporated as

$$\max_A \quad f_A + \frac{\alpha}{2M^2}\sum_{(i,j)(r,s)}\sum[\Omega_L(i,j) - \Omega_L(r,s)][\Omega_A(i,j) - \Omega_A(r,s)] \tag{2.2}$$

where M is the total number of bags, $2M^2$ is introduced as a normalizer to make α invariant to different number of bags, and the parameter α controls the trade-off between the bag constraints and the original clustering objective.

2.4 Incorporate Bag Constraints to Spectral Clustering

In this section, we incorporate the bag constraints into spectral clustering by modifying the *Normalized LinkRatio* objective. We show that this leads to a standard spectral clustering problem with a modified similarity matrix.

2.4.1 Preliminaries on Spectral Clustering

We first briefly review the spectral clustering. Let $A = [a_1, \cdots, a_K]$ be a *partition matrix*, where each column a_k is a binary assignment vector for cluster $\mathbb{X}_k$, with $a_{qk} = 1$ if instance x_q is assigned to cluster $\mathbb{X}_k$ and 0 otherwise. Let W be the symmetric *similarity matrix* of instances. Define the *degree matrix* $D = \text{Diag}(W\mathbf{1}_N)$, where $\text{Diag}(\cdot)$ forms a diagonal matrix with elements of the input vector as the diagonal elements, $\mathbf{1}_N$ denotes

a N-dimensional vector of all 1's, and N is the total number of vertices. The K-way spectral clustering with *Normalized LinkRatio* objective is defined as [120]

$$\max_{A} \quad \frac{1}{K}\sum_{k=1}^{K}\frac{a_k^T W a_k}{a_k^T D a_k} \tag{2.3}$$

$$\text{s.t.} \quad A \in \{0,1\}^{N\times K}, \quad A\mathbf{1}_K = \mathbf{1}_N. \tag{2.4}$$

Rewrite the objective as

$$\frac{1}{K}\sum_{k=1}^{K}\frac{a_i^T W a_i}{a_i^T D a_i} = \sum_{k=1}^{K}\frac{a_k^T D^{1/2} D^{-1/2} W D^{-1/2} D^{1/2} a_k}{a_k^T D a_k}.$$

Define $z_k = \frac{D^{1/2}a_k}{\|D^{1/2}a_k^T\|}$, and $Z = [z_1, \cdots, z_K]$. Ignoring the discrete constraint for Z at this stage, one can formulate a new clustering problem with respect to variable Z as

$$\max_{Z} \quad \text{tr}(Z^T D^{-1/2} W D^{-1/2} Z) \tag{2.5}$$

$$\text{s.t.} \quad Z^T Z = I \tag{2.6}$$

where the constraint (2.6) comes from the definition of Z. The solution of Z for this new problem is the eigenvectors associated with the K largest eigenvalues of $D^{-1/2}WD^{-1/2}$ [27]. Correspondingly, a discrete solution A of the original problem can be obtained by taking a rounding procedure from Z (e.g., using Kmeans or the approach proposed in [120]).

2.4.2 Spectral Clustering with Bag Constraints

To incorporate the bag constraints, we need to define the two similarity functions in (2.1), the class-label similarity function $\Omega_L(\cdot)$ and the cluster-label similarity $\Omega_A(\cdot)$. Ideally, $\Omega_L(\cdot)$ should satisfy the following conditions: (1) In the case where class label information between two bags $\mathbb{B}_i$ and $\mathbb{B}_j$ is unambiguous, (i.e., they do not share class label or they both belong to the same single class), $\Omega_L(i,j)$ should achieve minimum or maximum values; (2) In the ambiguous case where bags $\mathbb{B}_i$ and $\mathbb{B}_j$ have multiple labels and $\mathbb{Y}_i \cap \mathbb{Y}_j \neq \phi$, the smaller the quantity $\frac{|\mathbb{Y}_i \cap \mathbb{Y}_j|}{|\mathbb{Y}_i \cup \mathbb{Y}_j|}$ ($|\mathbb{Y}_i|$ is the number of distinct classes in $\mathbb{Y}_i$) is, i.e., the smaller the relative "common-label" set is, the smaller $\Omega_L(i,j)$ should

be.

Based on the above considerations, we define the following class-label similarity function. Let y_i be the $C \times 1$ binary class indicator vector for bag $\mathbb{B}_i$, with elements $y_{ic} = 1/|\mathbb{Y}_i|$ if $c \in \mathbb{Y}_i$, and $y_{ic} = 0$ otherwise. Denote $Y = [y_1, \cdots, y_M]$, where $y_m = \mathbf{0}$ for any bag $\mathbb{B}_m$ that is not labeled. The *class-label similarity* between (i, j) is defined as

$$\Omega_L(i, j) = y_i^T y_j \tag{2.7}$$

To define $\Omega_A(\cdot)$, denote the *bag indicator matrix* $B = [b_1, \cdots, b_M]$, with column vector $b_i \in \{0, 1\}^{N \times 1}$ and the element $b_{qi} = 1$ if instance $x_q \in \mathbb{B}_i$, and $b_{qi} = 0$ otherwise. The *cluster structure* of bag $\mathbb{B}_i$ can be captured by the $K \times 1$ column vector $Z^T D^{-1/2} b_i$. The k-th element in the cluster structure vector is $\frac{|\mathbb{X}_k \cap \mathbb{B}_i|}{\|D^{1/2} a_k^T\|}$, where $|\mathbb{X}_k \cap \mathbb{B}_i|$ counts the number of instances in bag $\mathbb{B}_i$ that belong to cluster $\mathbb{X}_k$. Essentially, $Z^T D^{-1/2} b_i$ forms a histogram of the cluster labels in bag $\mathbb{B}_i$ and normalizes each count by a quantity that can be roughly interpreted as the volume of the cluster.[3] This normalization allows the similarity measure to balance the contributions of clusters of different sizes. We now define the *cluster-label similarity* between (i, j) as

$$\Omega_A(i, j) = (Z^T D^{-1/2} b_i)^T (Z^T D^{-1/2} b_j) = b_i^T D^{-1/2} Z Z^T D^{-1/2} b_j \tag{2.8}$$

Substituting $\Omega_L(i, j)$ and $\Omega_A(i, j)$ into the inequality of bag constraints (2.1) , we have

$$\begin{aligned} &(y_i^T y_j - y_r^T y_s)(b_i^T D^{-1/2} Z Z^T D^{-1/2} b_j - b_r^T D^{-1/2} Z Z^T D^{-1/2} b_s) \geq 0 \\ &\quad \Leftrightarrow \operatorname{tr}(Z^T D^{-1/2} (y_i^T y_j - y_r^T y_s)(b_j b_i^T - b_s b_r^T) D^{-1/2} Z) \geq 0 \end{aligned}$$

where $y_i^T y_j - y_r^T y_s$ is a scalar. This inequality constraint is imposed for two pairs of bags. To incorporate the bag constraints for all pairs of bags, we follow the method introduced

[3]The normalization factor for cluster $\mathbb{X}_k$ is $\|D^{1/2} a_k^T\|$, where a_k is the binary indicator vector for cluster $\mathbb{X}_k$ and D is the degree matrix.

in (2.2) and add the following penalty term to the *Normalized LinkRatio* objective

$$\frac{\alpha}{2M^2}\sum_{(i,j)}\sum_{(r,s)}[\Omega_L(i,j)-\Omega_L(r,s)][\Omega_A(i,j)-\Omega_A(r,s)] \tag{2.9}$$

$$=\frac{\alpha}{2M^2}\mathrm{tr}(Z^TD^{-1/2}\sum_{(i,j)}\sum_{(r,s)}(y_i^Ty_j-y_r^Ty_s)(b_jb_i^T-b_sb_r^T)D^{-1/2}Z) \tag{2.10}$$

$$=\alpha\cdot\mathrm{tr}\left(Z^TD^{-1/2}B(Y^TY-\mu I)B^TD^{-1/2}Z\right) \tag{2.11}$$

$$=\alpha\cdot\mathrm{tr}\left(Z^TD^{-1/2}QD^{-1/2}Z\right),$$

$$\text{with}\ \ \mu=\frac{\mathbf{1}^\top(Y^TY)\mathbf{1}}{M^2}\ \ \text{and}\ \ Q=B(Y^TY-\mu I)B^T. \tag{2.12}$$

The two summations in (2.9) sum over all possible configurations of (i,j) and (r,s), and $\mathbf{1}^\top(Y^TY)\mathbf{1}$ in (2.12) sums over all the elements of Y^TY. The detailed derivation from (2.10) to (2.11) can be found in Appendix A.

Adding the above bag constraints as a penalty term to the *Normalized LinkRatio* objective (2.5) and taking into account the original constraint (2.6), we can rewrite the spectral clustering with bag constraints as

$$\max_Z \quad tr(Z^TD^{-1/2}(W+\alpha Q)D^{-1/2}Z) \tag{2.13}$$

$$\text{s.t.} \quad Z^TZ=I \tag{2.14}$$

It is easy to see that this formulation is equivalent to the standard spectral clustering (2.5) and (2.6) with a modified similarity matrix. We can then apply the general approach of spectral clustering to solve this optimization problem.

The spectral clustering algorithm with bag constraints is summarized in Algorithm 1. Note that in step 1, one can choose any method to compute the similarity matrix W so that the data similarities are properly captured (Existing methods include the ones in [77, 91, 122]). We applied the Kmeans rounding procedure in Step 6. One can, of course, apply any other appropriate rounding procedure. Step 2 involves several matrix multiplications. Since the dimension of Y is $C\times M$ and B is $N\times M$, the complexity of Step 2 is dominated by $\mathcal{O}(N^2M)$. Step 4 computes the top K eigenvectors of W', which is the most computationally expensive part. Using the Lanczos method, the complexity of Step 4 is $\mathcal{O}(KTnnz(W'))$, where T is the number of Lanczos iteration steps and $nnz(W')$

Algorithm 1 Spectral Clustering with MIML Bag Constraints

Input: A set of bags $\{\mathbb{B}_i\}_{i=1}^{M}$, $\mathbb{B}_i = \{x_{i1}, \cdots, x_{in_i}\}$; a set of known label sets associated with bags $\{(\mathbb{Y}_i, \mathbb{B}_i)\}$; parameter α; the number of instance clusters K.
Output: Instance clustering result.

1: Create instance similarity matrix $W \in \mathcal{R}^{N\times N}$; form the diagonal degree matrix $D = \mathrm{Diag}(W\mathbf{1}_N)$.
2: Form the label indicator matrix Y and the bag indicator matrix B, as described in Sec. 2.4. Construct the bag-constraint matrix $Q = B(Y^TY - \mu I)B^T$.
3: Compute the normalized similarity matrix with bag constraints $W' = D^{-1/2}(W + \alpha Q)D^{-1/2}$.
4: Find the K largest eigenvectors of $W', v_1, \cdots, v_K$; form the matrix $V = [v_1, \cdots, v_K] \in \mathcal{R}^{N\times K}$.
5: Re-normalize the rows of V to have unit length yielding $V' \in \mathcal{R}^{N\times K}$, i.e., $V'_{ij} = V_{ij}/(\sum_j V_{ij}^2)^{1/2}$.
6: Treat each row of V' as a point in $\mathcal{R}^K$ and cluster V' via Kmeans. Assign the original instance x_q to cluster $\mathbb{X}_k$ if and only if the q-th row of V' is assigned to $\mathbb{X}_k$.

is the number of nonzero elements in matrix W' [51]. Hence, the overall complexity is not increased by introducing bag constraints.

2.4.3 Relation to ML/CL Pairwise Constraints

As analyzed previously, in some cases pairwise constraints can be induced from the bag-level labels. When $K = C$, partitioning instances into K clusters is similar to predicting the class labels for instances. In this case, if a bag only has a single label then all instances within the bag belong to the same cluster and thus ML constraints can be imposed. Similarly, if two single-label bags have the same label, ML constraints should also be imposed on all pairs of instances formed across the two bags. For two bags that do not share any label, since they can not belong to the same cluster, CL constraints can be imposed on any instance pairs formed across the two bags. When $K > C$, some classes may correspond to more than one clusters. Thus, we can not impose ML constraints even for instances pairs that come from a single class label. However, CL constraints are still possible when two bags do not share any label.

The bag-constraint matrix Q introduced in Sec. 2.4 has some important properties that are closely related to pairwise constraints. We summarized these properties in the following proposition.

Table 2.1: Relation of Bag Constraints and Pairwise Constraints for $K = C$.

Cases	$Q_{p,q}$	ML/CL
$\|\mathbb{Y}_i\| = 1,\ x_p, x_q \in \mathbb{B}_i$	$1 - \mu$	ML
$\mathbb{Y}_i = \mathbb{Y}_j,\ \|\mathbb{Y}_i\| = \|\mathbb{Y}_j\| = 1,\ x_p \in \mathbb{B}_i,\ x_q \in \mathbb{B}_j$	1	ML
$\mathbb{Y}_i \cap \mathbb{Y}_j = \phi,\ x_p \in \mathbb{B}_i,\ x_q \in \mathbb{B}_j$	0	CL
$\|\mathbb{Y}_i\| > 1,\ x_p, x_q \in \mathbb{B}_i$	$\frac{1}{\|\mathbb{Y}_i\|} - \mu$	N/A
$\mathbb{Y}_i \cap \mathbb{Y}_j \neq \phi, \|\mathbb{Y}_i\| > 1$ or $\|\mathbb{Y}_j\| > 1, x_p \in \mathbb{B}_i, x_q \in \mathbb{B}_j$	$(0, 1)$	N/A

Proposition 1 (Properties of Q)**.** *Let $\mathbb{Y}_i$ and $\mathbb{Y}_j$ be the sets of class labels for bag $\mathbb{B}_i$ and bag $\mathbb{B}_j$ respectively. Let $|\mathbb{Y}_i|$ and $|\mathbb{Y}_j|$ be the sizes of the label set $\mathbb{Y}_i$ and $\mathbb{Y}_j$, respectively. Denote $Q_{p,q}$ as the value of the entry in Q that corresponds to the pair of instances x_p and x_q. Then the value of $Q_{p,q}$ can be determined according to Table 2.1.*

Proof. By the definition of Q in (2.12), we know that if $x_p, x_q \in \mathbb{B}_i$, $Q_{p,q} = \frac{1}{|\mathbb{Y}_i|} - \mu$, and that if $x_p \in \mathbb{B}_i, x_q \in \mathbb{B}_j$, $Q_{p,q} = \frac{|\mathbb{Y}_i \cap \mathbb{Y}_j|}{|\mathbb{Y}_i| \cdot |\mathbb{Y}_j|}$. It is thus easy to verify the first four cases. For the last case, since $|\mathbb{Y}_i \cap \mathbb{Y}_j| \neq \phi$, it follows that $|\mathbb{Y}_i \cap \mathbb{Y}_j| > 0$. Because the denominator $|\mathbb{Y}_i| \cdot |\mathbb{Y}_j|$ is also positive, we know $Q_{p,q} > 0$. Also given that $|\mathbb{Y}_i| > 1$ or $|\mathbb{Y}_j| > 1$, we know $|\mathbb{Y}_i \cap \mathbb{Y}_j| < |\mathbb{Y}_i| \cdot |\mathbb{Y}_j|$. Hence, $Q_{p,q} < 1$. □

It can be seen that, when ML constraints can be inferred for x_p and x_q, the value of $Q_{p,q}$ is 1 or $1 - \mu$ (approximately equal to 1 since μ is usually very small), which is the maximum of the constraint matrix. The value of $Q_{p,q}$ reaches 0 when CL constraints can be inferred. In other cases where some overlap exists between the class labels of two bags and no ML or CL can be imposed, the value of $Q_{p,q}$ lies in range $(0, 1)$ and the magnitude depends on the extend of the overlap. The more overlap their label sets have, the larger the value of $Q_{p,q}$ is. As such, we can view pairwise ML and CL constraints as only able to accommodate the cases where $Q_{p,q}$ takes extreme values. In contrast, our proposed method can capture different levels of ambiguity by allowing $Q_{p,q}$ to take a continuous value between zero and one, which potentially leads to more effective usage of the bag-level label information.

Table 2.2: Summary of MIML Datasets Information. Single-Label bags:number of bags that contain only a single class; Multi-Label bags: number of bags that have multiple labels; Avg. Inst.: average number of instances in each bag; Avg. Bag Label: average number of class labels in each bag.

Dataset	Birdsong	MSRC v2	Carroll	Frost
Classes	13	23	24	24
Dimension	38	48	16	16
Single-Label Bags	199	130	1	12
Multi-Label Bags	349	461	165	132
Total Bags	548	591	166	144
Total Inst.	4998	1758	717	565
Avg. Inst.	9.12	2.97	4.32	3.92
Avg. Bag Label	2.02	2.51	3.93	3.60

2.5 Empirical Evaluation

We conduct experiments on synthetic and real-world MIML datasets to evaluate the proposed bag-constrained spectral clustering method. The baseline methods include both unconstrained spectral clustering and existing spectral clustering algorithms with pairwise constraints.

2.5.1 Datasets Description

We use two real-world datasets and two synthetic datasets to evaluate our method. These datasets are previously used by a recent study on instance annotation for MIML [16]. The summary of the datasets is provided in Table 2.2.

HJA Birdsong is a real-word MIML dataset with 548 bags, each representing the spectrogram of a 10-second birdsong recording. Each instance corresponds to a bird song syllable in the spectrogram described by a 38-dimensional feature vector. There are 10232 instances, 4998 of which are provided with ground-truth class labels. For evaluation purpose, we use the filtered dataset, which only contains the labeled instances in each bag. Note that the ground-truth instance labels are only used in the evaluation.

MSRC v2 is the second version (v2) of Microsoft Research Cambridge (MSRC) image dataset,[4] containing 591 images and 23 classes. Each image is considered as a bag and regions in the images are viewed as instances. Each instance is described by a

[4] `http://research.microsoft.com/en-us/projects/objectclassrecognition/`

16-dimensional histogram of gradients and a 32-dimensional histogram of colors.

Letter-Carroll and **Letter-Frost** are two synthetic datasets generated using the Letter Recognition dataset from the UCI Machine Learning repository[5] and two poems.[6] To generate these datasets, words in the poems are viewed as bags and letters in each word are the instances and are randomly sampled (without replacement) from the Letter Recognition dataset. Bag-level labels are formed as the union of all letter labels in the word.

All datasets are standardized such that the mean of each feature is 0 and the standard deviation is 1. Instance similarities are computed using the *local scaling* factor proposed in [122]. Specifically, the similarity between instances x_p and x_q is computed by $W_{pq} = \exp\left(-\frac{\|x_p - x_q\|^2}{2\sigma_p\sigma_q}\right)$, where σ_p and σ_q are *local scaling* factors. The local scaling factor σ_q is defined as $\sigma_q = \|x_q - x_q^{(t)}\|$, where $x_q^{(t)}$ is the t-th nearest neighbor of x_q. We adopt $t = 7$ as recommended in [122], which is also shown to be effective in [95].

2.5.2 Baseline Methods

Our baseline methods include unconstrained spectral clustering (SP)and two constrained spectral clustering algorithms, Spectral Learning (*SpLearn*) algorithm proposed in [53] and constrained spectral clustering by regularization (*SpReg*) method proposed in [51]. SpLearn incorporates ML and CL constraints by directly modifying the entries of similarity matrix to 1 for ML constraints and to 0 for CL constraints. SpReg encodes ML constraints by adding a penalty term into the Normalized Cut objective. Note that SpReg only incorporates ML constraint but not CL constraints. To apply unconstrained spectral clustering (SP) to MIML instance clustering, we ignore the bag structure as well as the bag-level labels. For constrained spectral clustering, we create ML and CL constraints according to Table 2.1. The parameter β in SpReg that controls the enforcement of constraints is set to 20, the same value as that used in [51].

[5] `http://archive.ics.uci.edu/ml/machine-learning-databases/letter-recognition`

[6] The poem that generates the Letter-Carroll dataset is "*Jabberwocky*" written by Lewis Carroll in his 1872 novel *Through the Looking-Glass, and What Alice Found there*. The other poem that is used to create the Letter-Frost dataset is "*The Road Not Taken*" by *Robert Frost*, published in 1916 in the collection *Mountain Interval*.

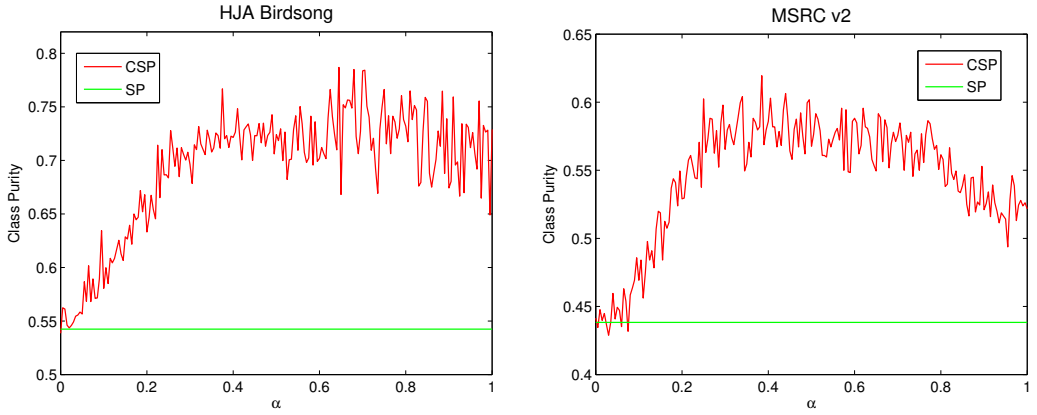


Figure 2.1: Class Purity results as a function of α. SP is unconstrained spectral clustering. CSP is the proposed spectral clustering with bag constraints.

2.5.3 Parameter Selection

In our algorithm, the parameter α is introduced to balance the trade-off between instance-feature similarity and the bag constraints. A large value of α imposes stronger restriction on the clustering solution to conform to the bag constraints and a small value of α produces clustering results without being heavily influenced by such constraints. We tested the performance of our method over a range of α values (from 0 to 1, by a 0.005 increment) on all our datasets and the results have shown that a value in the range $[0.5, 1]$ typically leads to significantly improved clustering performance. Figure 2.1 shows the performance of our method on the two real-world datasets as a function of α. In all the following experiments, the parameter α is set to 0.7.

2.5.4 Experiments and Discussions

We conducted experiments in two different scenarios. In the first scenario K is set to C and the goal is to group the instances based on their classes. In the second scenario, we have $K > C$ and some classes are represented by more than one cluster. In both scenarios, we test our algorithm with two implementations in order to thoroughly evaluate its performance. The first implementation (CSP) is a direct implementation of Algo-

rithm 1. The second implementation (CSP w.o.clml) is designed to test how well our algorithm could perform if we ignore the information that can be captured by ML and CL constraints. For this implementation, we set the entries that correspond to ML or CL constraints in the bag-constraint matrix Q to 0 and leave the rest unchanged. The two implementations are identical otherwise.

2.5.4.1 Scenario 1: $K = C$

In this scenario, we evaluate the performance of our method by changing the percentage of labeled bags. In particular, we vary the percentage of labeled bags from 20% to 100% of the whole dataset, with a 20% increment. For a fixed percentage, we randomly subsample bags (without replacement) to create the bag-constraint matrix and pairwise ML/CL constraints. The experiment is repeated for 20 random runs and the results are averaged.

We use two criteria to evaluate the clustering performance, *Normalized Mutual Information* (NMI) and *Class Purity*. The NMI is defined as

$$NMI = \frac{2I(\mathbf{X};\mathbf{C})}{H(\mathbf{X}) + H(\mathbf{C})} \tag{2.15}$$

where $\mathbf{X}$ and $\mathbf{C}$ are the numerical cluster and class label vectors, $I(\cdot;\cdot)$ computes the mutual information, and $H(\cdot)$ calculates the entropy. To compute *Class Purity*, each cluster is assigned to the most frequent class in the cluster, and then the accuracy of this assignment is measured by comparing the assigned labels with the ground-truth class labels. Formally,

$$\text{purity}(\mathbb{X},\mathbb{C}) = \frac{1}{N}\sum_{k}\max_{j}|\mathbb{X}_k \cap \mathbb{C}_j| \tag{2.16}$$

where $\mathbb{X} = \{\mathbb{X}_1,\ldots,\mathbb{X}_K\}$ is the set of clusters and $\mathbb{C} = \{\mathbb{C}_1,\ldots,\mathbb{C}_C\}$ is the set of classes.

The NMI and Class Purity results are reported in Fig. 2.2 and Fig. 2.3, respectively. From these results, we have the following observations and conclusions:

- Both CSP and CSP w.o.clml *outperforms* SP significantly as more bags are labeled.
- Our method is *comparable* with SpLearn and *outperforms* SpReg when the average

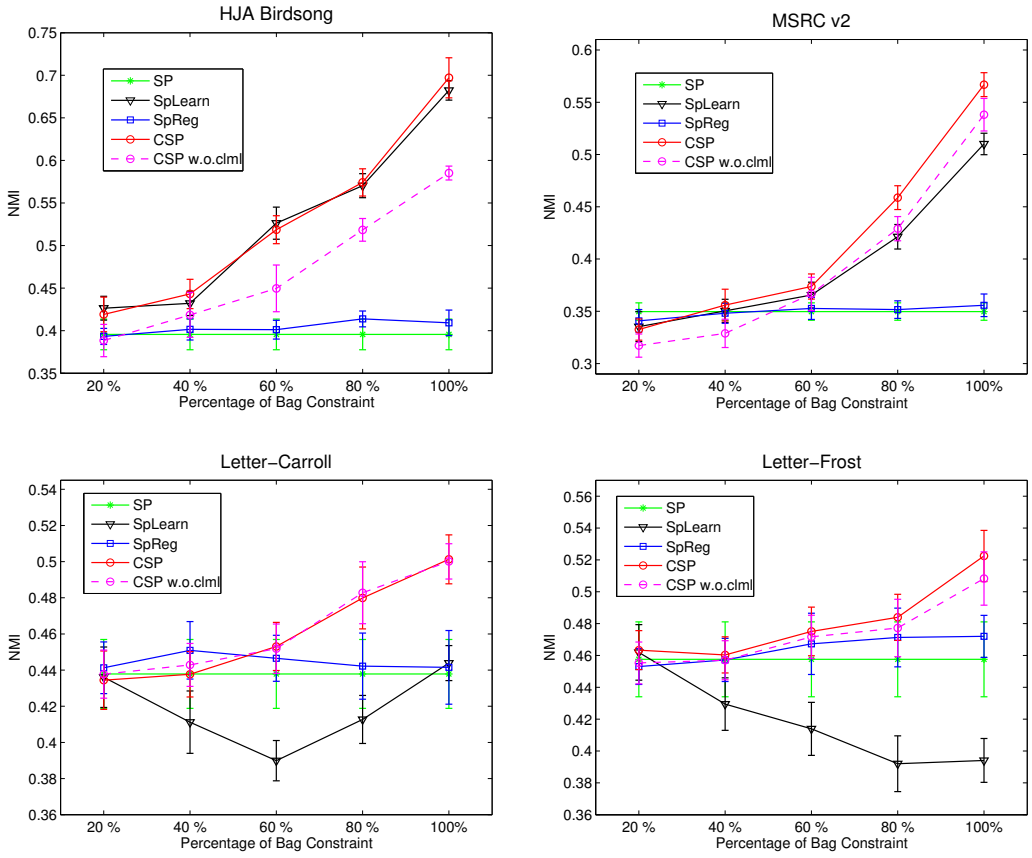


Figure 2.2: Scenario of $K = C$: NMI results as a function of constraints creating from different percentage of labeled bags. Error bars are reported with mean and standard deviation.

number of bag-level labels is *small* (HJA Birdsong). In this case, the ambiguity of instance labels induced from the bag-level labels is low, and many ML and CL constraints can be inferred. Such constraints can be properly incorporated by SpLearn to improve clustering. However, the number of ML constraints is relatively smaller compared to that of CL constraint, and thus the constraints do not help SpReg as much. In the meantime, the proposed method enables the clustering algorithm to incorporate information beyond what can be captured by the basic ML and CL constraints, which allows it to achieve competitive performance compared

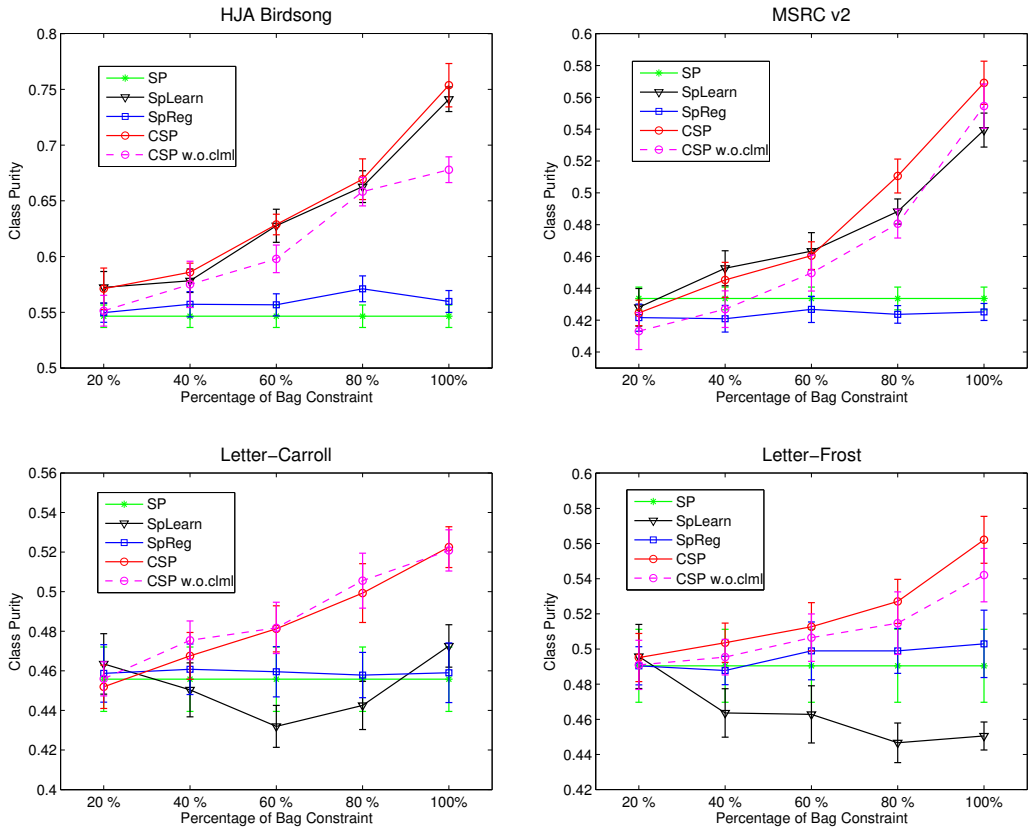


Figure 2.3: Scenario of $K = C$: Class Purity results as a function of different number of constraints. Error bars are reported with mean and standard deviation.

to SpLearn and SpReg.

- Our method *outperforms* SpLearn and SpReg when the number of average bag-level labels is relatively *larger* (MSRC v2, Letter-Carroll and Letter-Frost). In this case, very limited ML and CL constraints can be inferred, and our method with bag constraints better captures the side information in the bag-level labels. The fact that CSP and CSP w.o.clml performs almost the same in Letter-Carroll datasets indirectly demonstrates that ML and CL constraints can hardly be inferred. This gives more explanation why our method outperforms SpLearn and SpReg.

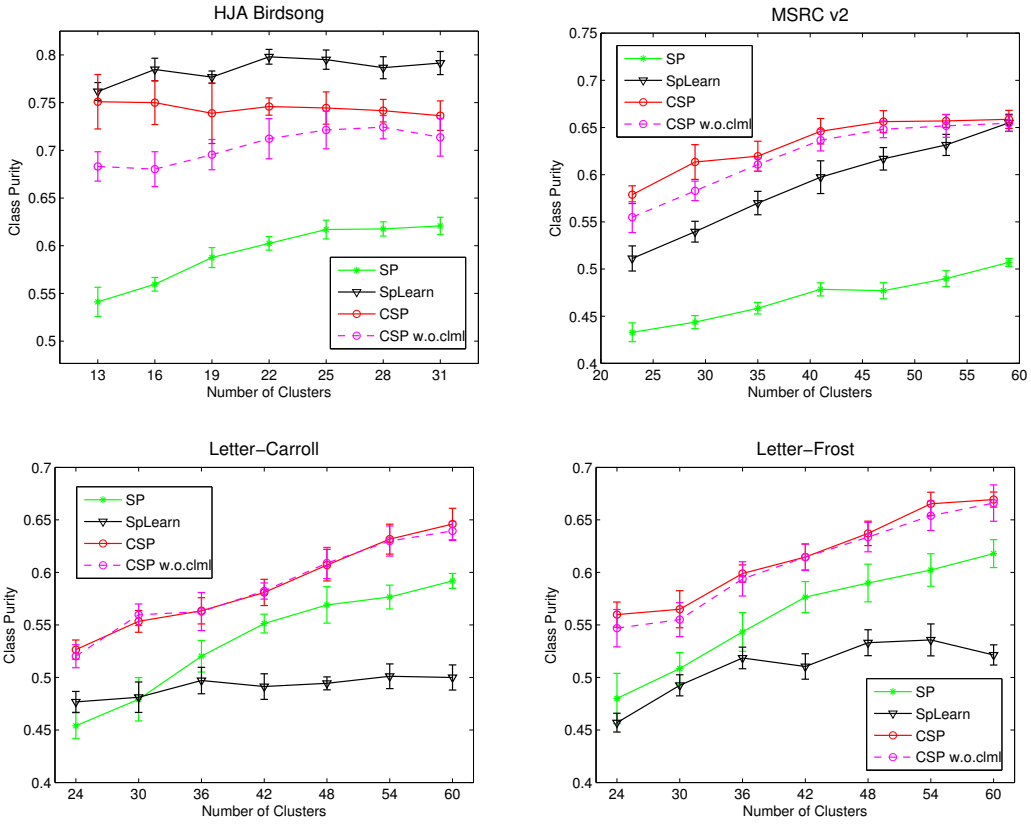


Figure 2.4: Scenario of $K > C$: Class Purity results as a function of the number of clusters. Error bars are reported with mean and standard deviation.

- On Letter-Carroll and Letter-Frost, while our method still *outperforms* SP, SpReg shows *no gain* and SpLearn actually leads to *degraded* clustering performance. Similar negative results have been reported in [30], which showed that constraint sets generated based on the ground truth labels can sometimes lead to degraded clustering performance. Further examination on these two datasets indicates that their bag-labels mostly induce CL constraints, which cannot be used by SpReg (thus explaining its flat performance). Moreover, the degraded performance by SpLearn suggests that CL constraints alone might not provide good guidance for MIML instance clustering. It is interesting to note that prior research [29] has demonstrated that CL constraints can sometimes make the solution space overly

constrained, leading to more difficult clustering problem. This provides a possible explanation for the degraded performance of SpLearn.

2.5.4.2 Scenario 2: $K > C$

In the scenario of $K > C$, we evaluate the performance of our method by changing the number of clusters K. For each dataset, we assume that all bags are labeled at the bag-level and vary the number of clusters from $K = C$ to roughly $2C$ with 7 steps. In the case of $K > C$, ML constraints can not be extracted (see discussion in Sec. 2.4.3). Hence, no constraints can be incorporated into SpReg and only CL constraints could be incorporated into SpLearn. We therefore do not consider the SpReg baseline in this scenario and remove ML constraints in SpLearn. The parameters setting is the same with the previous experiment.

We report the averaged Class Purity results over 20 runs in Fig. 2.4. NMI results are highly similar, and thus omitted to avoid redundancy. We can see that our method still outperforms SP consistently and significantly.

In the datasets with large average number of class labels (MSRC v2, Letter-Carroll, and Letter-Frost), we again observe that CSP and CSP w.o.clml performs similarly, which shows that few CL or ML constraints could be extracted. This is one of the possible reasons that SpLearn can not compete with our method for these datasets. Nonetheless, When the average number of class labels is small (HJA Birdsong), SpLearn excels. One possible explanation is that the bag label information in this dataset is relatively unambiguous and many pairwise constraints can be extracted. While our method can handle ambiguous information much better, SpLearn deals with unambiguous information more directly. Note that when we remove the single-label bags in HJA Birdsong and conduct the same experiment, we observed that our method is comparable with SpLearn (the result is not reported due to space limit). These results suggest that our method is more suitable for MIML datasets containing large numbers of multi-label bags.

2.6 Conclusion

In this chapter, we introduce a novel instance clustering problem in the MIML framework, where the bag-level labels are used as side information to inform the clustering of

instances. The goal is to recover the classes or to discover subclasses within each class. Traditional constraint-based clustering methods can not fully leverage the knowledge provided by the bag-level class labels. In contrast, we present a simple yet effective principle that incorporates the bag-level label information as bag constraints. The proposed constraints can be readily integrated into any optimization-based clustering algorithm by adding a penalty term to the objective. In this chapter, we demonstrate how the bag constraints can be incorporated into spectral clustering and empirically validate its effectiveness on both synthetic and real-world MIML datasets. The results show that the proposed bag-constrained method for spectral clustering generally outperforms state-of-the-art spectral clustering algorithms that use pairwise ML and CL constraints and is most suitable for MIML datasets that contain relatively large number of multi-label bags.

Manuscript 2: Comparing Clustering with Pairwise and Relative Constraints: A Unified Framework

Yuanli Pei, Xiaoli Z. Fern, Teresa V. Tjahja, and Rómer Rosales

ACM Transactions on Knowledge Discovery from Data (TKDD)
Published by ACM, New York, NY, USA
Volume 11 Issue 2, December 2016, Article No. 22

Chapter 3: Comparing Clustering with Pairwise and Relative Constraints: A Unified Framework

Abstract

Clustering can be improved with the help of side information about the similarity relationships among instances. Such information has been commonly represented by two types of constraints: *pairwise* constraints and *relative* constraints, regarding similarities about instance pairs and triplets respectively. Prior work has mostly considered these two types of constraints separately and developed individual algorithms to learn from each type. In practice, however, it is critical to understand/compare the usefulness of the two types of constraints as well as the cost of acquiring them, which has not been studied before. This chapter provides an extensive comparison of clustering with these two types of constraints. Specifically, we compare their impacts both on *human users* that provide such constraints and on the *learning system* that incorporates such constraints into clustering. In addition, to ensure that the comparison of clustering is performed on equal ground (without the potential bias introduced by different learning algorithms), we propose a probabilistic semi-supervised clustering framework that can learn from either type of constraints. Our experiments demonstrate that the proposed semi-supervised clustering framework is highly effective at utilizing both types of constraints to aid clustering. Our user study provides valuable insights regarding the impact of the constraints on human users, and our experiments on clustering with the human-labeled constraints reveal that relative constraint is often more efficient at improving clustering.

3.1 Introduction

Clustering is the task of organizing instances such that similar instances are grouped together. It plays an increasingly important role in many applications, including image analysis (e.g. [38, 128]), document retrieval (e.g. [34, 103]), and bioinformatics (e.g. [64, 69]). Traditionally, clustering is an unsupervised method for data analysis, where no labeled data is provided. However, in many applications, it is possible to acquire side information that can help infer the underlying instance-to-cluster assignments. Such

knowledge has been expressed as instance-level *constraints* for clustering, a common and useful form that reveals similarity relationships among instances.

Instance-level constraints mainly fall into two categories: *pairwise* constraints and *relative* constraints. A pairwise constraint specifies *absolute* similarity relationship between a pair of instances. That is, given a pair of instances $\mathbf{x}_a$ and $\mathbf{x}_b$, a Must-link (ML) constraint is introduced if they are similar to each other, and a Cannot-link (CL) constraint is provided otherwise. Relative constraints, in comparison, reveal *comparative* similarity relationships among instance triplets $\mathbf{x}_a$, $\mathbf{x}_b$, and $\mathbf{x}_c$, i.e., each constraint specifies whether instance $\mathbf{x}_a$ *is more similar to* $\mathbf{x}_b$ *than to* $\mathbf{x}_c$. Both types of constraints have been previously incorporated into clustering using metric learning approaches (e.g., [2, 22, 31, 66, 83, 89, 110]), or methods that directly search for clustering solutions that respect the constraints (e.g., [7, 80, 90, 102]).

Despite the broad study on both types of constraints, most work has focused on only one type. Moreover, most work assumes that the constraints are given beforehand and they only focus on learning from the constraints without considering the cost of obtaining them. However, to understand which type of constraints is more suitable for practical applications, it is necessary to compare their effectiveness in improving clustering as well as the ease of obtaining reliable constraints, which has not been studied before. This chapter takes into account both aspects and provides an extensive comparison of clustering with these two types of constraints. Specifically, we aim to provide answers to the following questions.

Q1: Given the same amount of labeling effort, which type of constraint is more effective at aiding clustering?

Earlier work on relative constraints [58, 80, 83] compared their approaches with existing methods that use pairwise constraints and often showed superiority. However, it is difficult to draw a strong conclusion from such comparisons because they used different learning algorithms for different types of constraints and it is not clear whether the performance gain they reported is due to the use of different learning algorithms or the use of different types of constraints.

Q2: Which type of constraints is easier to obtain from human labelers?

On the surface, labeling pairwise constraints may seem easier as there are fewer instances to be examined in each query. However, providing pairwise constraints requires

absolute judgement (similar or dissimilar) that has long been believed to be more difficult to make than the comparative judgement needed by relative constraints [79]. Therefore, it is difficult to answer this question reliably without more thorough experimentation.

Q3: Which type of constraint is more reliable (i.e., the labels are more accurate)?

This question concerns the quality of the constraints. Generally, we cannot guarantee noise-free constraints from human labelers due to various uncertainties in the labeling process. The question is then which type is less noisy if other factors are controlled. Research in psychology reveals that humans are often inaccurate in making absolute judgments, but they are more trustworthy when judging comparatively [79]. This may imply that relative constraints provide more accurate information. But there are more instances to consider when labeling relative constraints, leading to higher chance of making an incorrect judgment on individual instances, which may, in turn, result in a higher possibility of mistakes in labeling relative constraints.

To answer Q1, we first need a clustering method that can incorporate both types of constraints in the same fashion to avoid potential comparison bias introduced by the learning algorithms. Toward this end, we first propose a unified semi-supervised clustering framework (Section 3.2). The framework uses a probabilistic model to describe the relationships between instances, their underlying cluster memberships, and the observed constraints (Section 3.2.2). Based on the model, we present an objective that can be used for clustering with either pairwise or relative constraints (Section 3.2.3). To optimize the objective, we develop a variational Expectation-Maximization (EM) solution (Section 3.2.4). We then theoretically analyze the *informativeness* of the two types of constraints (Section 3.4) and empirically compare the clustering performance with them (Section 3.5). To answer Q2 and Q3, we need to investigate how human labelers behave during the labeling process. In this work, we report a user study (Section 3.5) that we have designed and conducted specifically to answer these questions.

Our experiments with synthetically generated constraints (Section 3.3) demonstrate that the proposed semi-supervised clustering framework is highly effective at utilizing both types of constraints to aid clustering. Our user study (Section 3.5) empirically compares the two types of constraints from the perspectives of their impact on users (regarding Q2 and Q3) and their effect on learning systems (regarding Q1).

3.2 A Unified Framework for Clustering with Constraints

In this section, we introduce our unified probabilistic model for clustering with instance-level constraints.

3.2.1 Problem Statement

In the problem of clustering with constraints, we are given a dataset along with a set of constraints on the instances. Let $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T$ be the given data, where each $\mathbf{x}_i \in \mathcal{R}^d$, and d is the dimension of the feature space. Let $Y = [y_1, \ldots, y_N]^T$ be the hidden cluster label vector, where y_i is the unknown cluster label of $\mathbf{x}_i$. Below we will use superscripts p and r to distinguish the notations between pairwise and relative constraints and omit the superscripts when the discussion applies to both cases. We denote the index set of constraints as $\mathcal{C} = \{I_1, \ldots, I_M\}$, where $I_j^p = (a_j, b_j)$ and $I_j^r = (a_j, b_j, c_j)$ contains the instance indices in the j-th pairwise or relative constraint. Correspondingly, we denote the cluster labels of the instances involved in the j-th constraint as $Y_{I_j}^p = [y_{a_j}, y_{b_j}]$ and $Y_{I_j}^r = [y_{a_j}, y_{b_j}, y_{c_j}]$. Below we use I to index all the distinct instances involved in the constraints, i.e., $I = \left\{1 \le i \le N : i \in \cup_{j=1}^{M} I_j\right\}$[1], and denote U as the set of indices for all the instances that are not in any constraint.

In this chapter, we consider a simple and practical setting where both types of constraints are obtained by querying the human labelers and the answers are provided based on the perceived underlying classes/clusters. Specifically, for a pairwise constraint, we query the labeler *whether two instances* $\mathbf{x}_a$ *and* $\mathbf{x}_b$ *are similar or not*, and the constraint label $l^p \in \{ML, CL\}$ is provided based on

$$l^p = \begin{cases} ML, & y_a = y_b \\ CL, & y_a \neq y_b \ . \end{cases} \tag{3.1}$$

For a relative constraint, we ask the labeler to indicate, among the triplet $(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$, *which two instances are more similar than the other*. The constraint label l^r is returned

[1]Notice the subtle difference between the notation $\mathcal{C}$ and I. While both of them contain indices of instances in the constraints, $\mathcal{C}$ retains the instance ordering in each constraint especially for relative constraints, while I only records the distinct instances in all the constraints.

based on

$$l^r = \begin{cases} ab\bar{c}, & y_a = y_b,\ y_a \neq y_c \\ a\bar{b}c, & y_a = y_c,\ y_a \neq y_b \\ \bar{a}bc, & y_a \neq y_b,\ y_b = y_c \\ ow, & o.w. \end{cases} \tag{3.2}$$

where the *ow* (otherwise) label corresponds to the case where it is not possible for the labeler to make a reasonable decision (for example, if the three instances are equally similar or dissimilar to each other) [2]. Note that in the notation of the constraint label $l^r \in \{ab\bar{c}, a\bar{b}c, \bar{a}bc, ow\}$, we use $\bar{\cdot}$ to indicate the less similar instance, e.g., $l^r = ab\bar{c}$ means that $\mathbf{x}_a$ is more similar to $\mathbf{x}_b$ than to $\mathbf{x}_c$. We denote the vector of the observed constraint labels as $L = [l_1, \ldots, l_M]^T$ for both types of constraints.

In this problem, given the data X, a set of constraints $\mathcal{C}$, and the constraint labels L, our goal is to *partition the data X into K disjoint clusters such that similar instances are grouped together guided by the given pairwise or relative constraints.* In this chapter, we assume that K is pre-specified. The choice of K needs extensive study and the discussion is out of the scope of this chapter.

3.2.2 The Probabilistic Model

Figure 3.1(a) shows the unified graphical model defining the independence assumptions between input instances $\mathbf{x}$'s, their cluster labels y's and the observed constraint labels l's. In the model, the $\mathbf{x}_i$'s are i.i.d; the distributions of y_i's are determined by $\mathbf{x}_i$ and the parameter W; the l's only depend on the y's. In general, l can be the label of any instance-level constraints. When instantiating with pairwise and relative constraints, l's are only related to the cluster labels of the instances involved in the pair or triplet. Thus, the model can be further simplified as shown in Figure 3.1(b) and 3.1(c).

Given the model, for both pairwise and relative constraints, the joint distribution

[2] Although our setting of relative constraints is similar with [80], the constraint labels are defined differently. [80] assumes that each constraint is obtained by querying: *is $\mathbf{x}_a$ more similar to $\mathbf{x}_b$ than to $\mathbf{x}_c$*. Due to the nature of the query, they did not explicitly model constraints with label $\bar{a}bc$. Rather, it is included in the *dnk* case (corresponding to *ow* here). In contrast, here we explicitly model the $\bar{a}bc$ case.

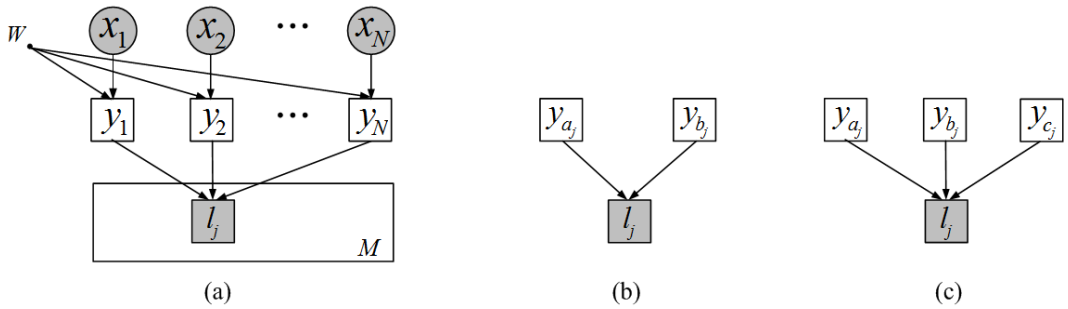


Figure 3.1: The unified graphical model for clustering with constraints. (a) the unified model; (b) instantiation of pairwise constraints; (c) instantiation of relative constraints. For simplicity, only the variables directly connected with a single constraint label l_j are shown. When multiple labels are considered the directed graph normally has loops and exact inference is not tractable.

consistent with the graphical model can be factorized as

$$P(L, Y|X; W) = P(L|Y)P(Y|X; W) = \prod_{j=1}^{M} P(l_j|Y_{I_j}) \prod_{i=1}^{N} P(y_i|\mathbf{x}_i; W), \tag{3.3}$$

Below we describe our design for the conditional distributions $P(y_i|\mathbf{x}_i; W)$ and $P(l_j|Y_{I_j})$. Our following discussion applies to both types of constraints. Unless it is necessary, we will not distinguish between the two.

We first use a multi-class logistic classifier [14] to model the conditional probability $P(y_i|\mathbf{x}_i; W)$. Let $W = [w_1, \ldots, w_K; b_1, \ldots, b_K]^T$ be a weight matrix in $\mathcal{R}^{K\times(d+1)}$, where each $w_k \in \mathcal{R}^d$ contains weights on the feature space and b_k's are the bias terms. The conditional probability $P(y_i|\mathbf{x}_i; W)$ is defined as

$$P(y_i = k|\mathbf{x}_i; W) = \frac{\exp\left(w_k^T \mathbf{x}_i + b_k\right)}{\sum_{k'} \exp\left(w_{k'}^T \mathbf{x}_i + b_{k'}\right)}, \ \forall i, \ k.$$

We then define the distribution of the constraint labels given the clusters labels of the involved instances $P(l_j|Y_{I_j})$. In an ideal scenario where the given constraints are always correct, the distribution of $l_j|Y_{I_j}$ is deterministic, namely, the label l_j is deterministic given the assignments of Y_{I_j}. In practice, however, labelers can make mistakes and be inconsistent in labeling. To address this issue, we relax the deterministic relationship

Table 3.1: Pairwise constraints: $P(l_j^p|Y_{I_j}^p)$ with $Y_{I_j}^p = [y_{a_j}, y_{b_j}]$.

Cases	$l_j^p = ML$	$l_j^p = CL$
$y_{a_j} = y_{b_j}$	$1-\epsilon$	ϵ
$y_{a_j} = y_{c_j}$	ϵ	$1-\epsilon$

Table 3.2: Relative constraints: $P(l_j^r|Y_{I_j}^r)$ with $Y_{I_j}^r = [y_{a_j}, y_{b_j}, y_{c_j}]$.

Cases	$l_j^r = ab\bar{c}$	$l_j^r = a\bar{b}c$	$l_j^r = \bar{a}bc$	$l_j^r = ow$
$y_{a_j} = y_{b_j}, y_{a_j} \neq y_{c_j}$	$1-\epsilon$	$\epsilon/3$	$\epsilon/3$	$\epsilon/3$
$y_{a_j} = y_{c_j}, y_{a_j} \neq y_{b_j}$	$\epsilon/3$	$1-\epsilon$	$\epsilon/3$	$\epsilon/3$
$y_{b_j} = y_{c_j}, y_{a_j} \neq y_{b_j}$	$\epsilon/3$	$\epsilon/3$	$1-\epsilon$	$\epsilon/3$
o.w.	$\epsilon/3$	$\epsilon/3$	$\epsilon/3$	$1-\epsilon$

for $P(l_j^p|Y_{I_j}^p)$ and $P(l_j^r|Y_{I_j}^r)$ as described in Table 3.1 and 3.2 respectively. The relaxation is parameterized by $\epsilon \in [0, 1)$, indicating the probability of an error when labeling the constraints. Take relative constraints for example, if $y_{a_j} = y_{b_j}, y_{a_j} \neq y_{c_j}$, then the constraint label is returned as $l_j^r = ab\bar{c}$ in the deterministic relations. In our relaxation distribution instead, the ideal label $l_j^r = ab\bar{c}$ is given with probability $1-\epsilon$, and any other label among $a\bar{b}c$, $\bar{a}bc$ and ow is given with equal probability $\epsilon/3$. In this way, the relaxation accommodates the erroneous labels. Additionally, this relaxation allows the constraints to be *soft* as needed. Later in our objective, we will introduce a term to enforce cluster balance and separation. By allowing for such soft constraints, the relaxation balances the trade-off between satisfying all constraints and finding large separation margins among balanced clusters.

3.2.3 Objective

Our objective includes learning from both the constraints and the unlabeled instances. Firstly, to learn from the given constraints, we maximize the likelihood of the observed constraint labels given the instances, i.e.,

$$\Phi(L|X_I; W) = \frac{1}{M} \log P(L|X_I; W) = \frac{1}{M} \log \sum_{Y_I} P(L, Y_I|X_I; W), \tag{3.4}$$

where I indexes the constrained instances as defined in Section 3.2.1, and $\frac{1}{M}$ is a normalization constant.

Secondly, we also learn from the unlabeled data using unsupervised clustering method. Generally, a good clustering solution desires large separation margins to achieve high confidences for cluster memberships of individual instances, and a balanced distribution across clusters to avoid degenerated solutions. Following this principle, we incorporate the unsupervised clustering objective in [44], which can be naturally combined with our model. This objective is to maximize the mutual information between instance features and the hidden cluster labels:

$$I(Y, X; W) = H[\hat{y}|W] - H[Y_U|X_U; W]. \tag{3.5}$$

The first term in (3.5) is the conditional entropy of the empirical cluster label distribution $H[\hat{y}|W] = -\sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k$, where $\hat{p}_k = \frac{1}{N} \sum_{i=1}^{N} p(y_i = k|\mathbf{x}_i; W)$. This entropy is maximized when the cluster labels are uniformly distributed, namely, the clusters are balanced. Since this entropy term regards the structure of the entire data, we use both constrained and unconstrained instances for this term.

The second term is the conditional entropy of instance cluster labels for the unlabeled instances $H[Y_U|X_U; W] = \frac{1}{|U|} \sum_{i \in U} H[P(y_i|\mathbf{x}_i; W)]$. When it is minimized, the formed clusters will have large separation margin and high confidence for the cluster memberships of unconstrained instances. Note that we only use unconstrained instances in this term, since stronger cluster membership information about the constrained instances has already been captured by (3.4).

Combining these two objectives above, and adding a regularization for the parameter W, our unified objective becomes

$$\max_{W} \ \Phi(L|X_I; W) + \tau I(Y, X; W) - \lambda R(W). \tag{3.6}$$

where $R(W) = \sum_k w_k^T w_k$ is the L-2 regularization, and λ and τ are coefficients.

3.2.4 Optimization

Now we present our approach that optimizes the objective (3.6) when either pairwise or relative constraints are given. Computing the likelihood term requires marginalizing over hidden variables Y_I. When many y's are related to each other via constraints, marginalization becomes too expensive to compute and exact inference may be intractable. For this reason, we use variational EM for optimization.

In variational EM, a lower bound of the objective is found and maximized alternately in the E-steps and M-steps [14]. The first term in the objective can be written as $\log E_{Q(Y_I)}\left[\frac{P(Y_I,L|X_I;W)}{Q(Y_I)}\right]$, where $Q(Y_I)$ is a variational distribution. Applying Jensen's inequality to the log function, we obtain the lower bound

$$LB = \frac{1}{M} E_{Q(Y_I)}\left[\log \frac{P(Y_I, L|X_I; W)}{Q(Y_I)}\right] + \tau I(Y, X; W) - \lambda R(W). \tag{3.7}$$

In each E-step, for a fixed parameter W, the LB is maximized when the distribution $Q(Y_I)$ minimizes the Kullback-Leibler divergence $KL[Q(Y_I)||P(Y_I|L, X_I; W)]$. Given the $Q(Y_I)$ found in the E-step, the M-step updates W to maximize LB. Note that in the objective (and LB), only the likelihood term is relevant to E-steps. The other terms are only related in solving for W in M-steps.

3.2.4.1 Variational E-Step

We use mean field inference [41, 86] to find the variational distribution $Q(Y_I)$, due to its ease of implementation and convergence properties [8]. Mean field method restricts $Q(Y_I)$ to the fully-factorized family $Q(Y_I) = \prod_{i \in I} q(y_i)$, and find the $Q(Y_I)$ that minimizes $KL[Q(Y_I)||P(Y_I|L, X_I; W)]$. The optimal $Q(Y_I)$ is obtained by iteratively updating each $q(y_i)$ until convergence. The update equation in each iteration is

$$q(y_i) = \frac{1}{Z} \exp\{E_{Q(Y_{I\setminus i})}[\log P(Y_I, L|X_I)]\}\ , \tag{3.8}$$

where $Q(Y_{I\setminus i}) = \prod_{u \in I, u \neq i} q(y_u)$, and Z is a normalization factor to ensure $\sum_{y_i} q(y_i) = 1$. Below we derive a closed-form update of (3.8) for our model.

Substituting the model factorization (3.3) to the expectation term in (3.8), we have

$$E_{Q(Y_{I\backslash i})}[\log P(Y_I, L|X_I)] = \sum_{j:i\in I_j} E_{Q(Y_{I_j\backslash i})}[\log P(l_j|Y_{I_j})] + \log P(y_i|\mathbf{x}_i; W) + c, \quad (3.9)$$

where c absorbs all the constant terms with respect to y_i, and $I_j\backslash i$ is the indices in I_j removing i [3]. The first term in (3.9) sums over the expected log-likelihood of observing each l_j given the fixed y_i. To compute the expectation, we first let $\tilde{Q}(l_j|y_i, I_j)$ be the probability that the observed constraint label l_j is *consistent* with the fixed y_i and the other assignments of $Y_{I_j\backslash i}$. That is, $\tilde{Q}(l_j|y_i, I_j)$ is the probability of the event that $Y_{I_j}|y_i$ satisfies the conditions to allow $P(l_j|Y_{I_j}) = 1 - \epsilon$ according to Table 3.1 or 3.2. For both pairwise and relative constraints, the values $\tilde{Q}(l_j^p|y_i, I_j^p)$ and $\tilde{Q}(l_j^r|y_i, I_j^r)$ can be computed straightforwardly as in Table 3.3 and 3.4. Then, each of the expectations in (3.9) is computed as

$$E_{Q(Y_{I_j\backslash i})}[\log P(l_j|y_i, I_j)] = [1 - \tilde{Q}(l_j|y_i, I_j)] \log \delta(\epsilon) + \tilde{Q}(l_j|y_i, I_j) \log(1 - \epsilon).$$

where $\delta(\epsilon) = \epsilon$ for pairwise constraints and $\delta(\epsilon) = \epsilon/3$ for relative constraints, representing the probability for the erroneous label.

With the above derivation, we can simplify the update equation (3.8) as

$$q(y_i) = \frac{\alpha^{F(y_i)} P(y_i|\mathbf{x}_i; W)}{\sum_{y_i} \alpha^{F(y_i)} P(y_i|\mathbf{x}_i; W)}, \quad \text{with} \quad F(y_i) = \sum_{j:i\in I_j} \tilde{Q}(l_j|y_i, I_j) \quad (3.10)$$

where $\alpha = (1 - \epsilon)/\epsilon$ for pairwise constraints and $\alpha = 3(1 - \epsilon)/\epsilon$ for relative constraints.

We can interpret the term $F(y_i)$ as a measurement of the compatibility of each y_i with respect to the constraints and the other cluster labels involved in the same constraints. The α in (3.10) is controlled by the parameter ϵ. When $\epsilon \in (0, \frac{1}{2})$ for pairwise constraints or $\epsilon \in (0, \frac{3}{4})$ for relative constraints, we have $\alpha > 1$ and the update (3.10) allows for more compatible assignments of y_i's (the ones with higher $F(y_i)$ values) to have larger $q(y_i)$. When $\epsilon = \frac{1}{2}$ for pairwise constraints or $\epsilon = \frac{3}{4}$ for relative constraints, the constraint labels are regarded as uniformly distributed regardless of the instance cluster labels, as can be seen from the distribution $P(l_j|Y_{I_j})$ in Table 3.1 and 3.2. In this case, $\alpha = 1$

[3] Note that $I_j\backslash i$ contains one instance for a pairwise constraint, and two instances for a relative constraint.

Table 3.3: Pairwise Constraints: the values of $\tilde{Q}(l_j^p|y_i = k, I_j^p)$, $i \in I_j^p$. For simplicity, we denote $q_{\cdot k} = q(y_{\cdot_j} = k)$.

Cases	$l_j^p = ML$	$l_j^p = CL$
$i = a_j$	q_{bk}	$1 - q_{bk}$
$i = b_j$	q_{ak}	$1 - q_{ak}$

Table 3.4: Relative Constraints: the values of $\tilde{Q}(l_j^r|y_i = k, I_j^r)$, $i \in I_j^r$. For simplicity, we denote $q_{\cdot k} = q(y_{\cdot_j} = k)$ and $q_{\cdot \bar{k}} = \sum_{u \neq k} q(y_{\cdot_j} = u)$.

Cases	$l_j^r = ab\bar{c}$	$l_j^r = a\bar{b}c$	$l_j^r = \bar{a}bc$	$l_j^r = ow$
$i = a_j$	$q_{bk}q_{c\bar{k}}$	$q_{b\bar{k}}q_{ck}$	$\sum_{u \neq k} q_{bu}q_{cu}$	$1 - q_{bk}q_{c\bar{k}} - q_{b\bar{k}}q_{ck} - \sum_{u \neq k} q_{bu}q_{cu}$
$i = b_j$	$q_{ak}q_{c\bar{k}}$	$\sum_{u \neq k} q_{au}q_{cu}$	$q_{a\bar{k}}q_{ck}$	$1 - q_{ak}q_{c\bar{k}} - q_{a\bar{k}}q_{ck} - \sum_{u \neq k} q_{au}q_{cu}$
$i = c_j$	$\sum_{u \neq k} q_{au}q_{bu}$	$q_{ak}q_{b\bar{k}}$	$q_{a\bar{k}}q_{bk}$	$1 - q_{ak}q_{b\bar{k}} - q_{ak}q_{b\bar{k}} - \sum_{u \neq k} q_{au}q_{bu}$

and each $q(y_i)$ is directly set to the conditional probability $P(y_i|\mathbf{x}_i; W)$. This naturally reduces our method to learning without constraints. Clearly, when ϵ is smaller, the constraints are harder and the updates will push $q(y_i)$ to more extreme distributions to favor assignments that are consistent with the constraints. Note that when $\epsilon \in (\frac{1}{2}, 1)$ for pairwise constraints or $\epsilon \in (\frac{3}{4}, 1)$ for relative constraints, the value $\alpha < 1$, which will lead to results that contradict the constraints, and such cases are generally not desired.

Special Case: Hard Constraints. In the special case where $\epsilon = 0$ and $\alpha = \infty$, $P(l_j|Y_{I_j})$ essentially reduces to the deterministic relations, allowing our model to incorporate *hard* constraints. The update equation of this case can also be addressed similarly. In this case, $q(y_i)$ is non-zero only when the assignment of y_i is the most consistent with the observed constraints (Note that there could be multiple maximum). Thus, the update equation is reduced to a *max* model. More formally, we define $\tilde{Y}_i$ as the max-compatible label set for $\mathbf{x}_i$ with respect to the constraints, namely,

$$\tilde{Y}_i = \{1 \leq k \leq K : F(y_i = k) = \max_{k'} F(y_i = k')\}.$$

Then the update equation in the mean field steps becomes

$$q(y_i) = \begin{cases} \frac{P(y_i|\mathbf{x}_i;W)}{\sum_{y'_i \in \tilde{Y}_i} P(y'_i|\mathbf{x}_i;W)}, & \text{if } \ y_i \in \tilde{Y}_i \ , \\ 0, & \text{o.w.} \end{cases} \tag{3.11}$$

3.2.4.2 M-Step

With the distribution $Q(Y_I)$ found in E-step, the M-step updates W to maximizes the LB. By applying the factorization (3.3) and simplifying the LB, we obtain the following objective

$$\max_{W} \ \frac{1}{M} \sum_{Y_I} Q(Y_I) \log P(Y_I|X_I;W) + \tau I(Y,X;W) - \lambda R(W).$$

This optimization can be solved via gradient ascent methods. In our experiments, we use the L-BFGS algorithm [87].

3.2.4.3 Complexity and Initialization

For each E-step, the complexity is $\mathcal{O}(\gamma K|I|)$, where γ is the number of mean-field iterations for $Q(Y_I)$ to converge. In the M-step, the complexity of computing the gradient of W in each L-BFGS iteration is $\mathcal{O}(NKd)$.

The mean-field approximation used in the E-step is guaranteed to converge. In practice, we can use a fixed number of update iterations as an extra termination condition. Our empirical results show that mean field converges very fast, especially in later EM iterations.

To better initialize our method, we first apply Kmeans and train a supervised logistic classifier with the clustering results. The learned weights are then used as the starting point of our method. Empirically we observe that such initialization typically allows the algorithm to converge within 100 iterations.

Table 3.5: Summary of Constrained Clustering Datasets Information.

Dataset	#Inst.	#Dim.	#Cluster
Ionosphere	351	34	2
Pima	768	8	2
Balance-scale	625	4	3
Digits-389	3165	16	3
Letters-IJLT	3059	16	4
MSRCv2	1046	48	6
Flower	1066	128	7
HJA Birdsong	4998	38	13
Birdsong v2	2467	38	13

3.3 Performance of the Proposed Clustering Framework

In this set of experiments, we show that the proposed framework is very effective at clustering with pairwise and relative constraints. This allows us to later compare the two types of constraints using an effective learning algorithm.

3.3.1 Datasets

We use five UCI datasets and four additional real-world datasets. The UCI datasets include: *Ionosphere*, *Pima*, *Balance-Scale*, *Digits-389*, and *Letters-IJLT*. The four additional datasets are: 1) a subset of the *MSRCv2* dataset [73], which contains image segments of six common classes; 2) a subset of the *Flower* data [78], which contains seven flower classes that are difficult to classify due to high within-class variances and between-class similarities; 3) the *HJA Birdsong* data [17], which contains segments automatically extracted from spectrograms of birdsong recordings and the goal is to identify the bird species that make the utterance for the corresponding segment; and 4) the *Birdsong v2* data [80], another birdsong dataset that contains manually inspected segments that are labeled with bird singing patterns. Here the task is to find distinct bird song patterns instead of the bird species (Note that each species may contain multiple unique song patterns). We summarize the dataset information in Table 3.5. In our experiments, all features of all datasets are standardized to have zero mean and unit standard deviation.

3.3.2 Baseline Methods and Evaluation Metric

We evaluate our algorithm using both pairwise and relative constraints. For pairwise constraints, we compare with three well-known methods: *Xing*'s method [110] (distance metric learning for a diagonal matrix), *ITML* [31], and *Constrained EM* [90]. Xing's method is one of the earliest methods learning from pairwise constraints. ITML is, based on our experience, a very effective and efficient method developed in recent years. Constrained EM is another probabilistic method that learns from pairwise constraints. It employs a generative model, which is different with the discriminative model used in our method.

For relative constraints, we compare with: 1) *LSML* [66], a distance metric approach (here Euclidean distance is used as the prior); 2) *SSSVaD* [58], a method that finds clustering solutions and learns a distance metric from the relative constraints; and 3) *sparseLP* [83], an earlier method that has not been extensively compared with. We also experimented with the SVM-style method in [89]. Its performance is generally worse than the other competing methods, and the result is not reported. Note that due to the design of these methods, they can only employ relative constraints with label $ab\bar{c}, a\bar{b}c$ or $\bar{a}bc$, and can not be easily extended to incorporate the *ow* constraints.

Several methods in the baselines incorporate constraints using metric learning techniques: Xing's method, ITML, LSML, and sparseLP. For such methods, we apply Kmeans (with 50 random restarts) using the learned metric to form the clustering results. We evaluate the clustering results of all methods using *pairwise F-measure* [13], *Adjusted Rand Index* and *Normalized Mutual Information*. The results are highly similar across different measures, and here we only present the F-Measure results.

3.3.3 Overall Performance

3.3.3.1 Experimental Setup

In this set of experiments, we evaluate all methods using synthetically generated constraints. We first randomly generate instance pairs or triplets and then assign the constraint labels deterministically based on their cluster labels. Namely, we label a pairwise constraint $(\mathbf{x}_a, \mathbf{x}_b)$ as ML if the cluster label y_a and y_b are the same, and CL otherwise. For relative constraints, we label the constraint $(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$ as $ab\bar{c}$ if y_a is the same with

y_b and different with y_c, and similarly for the case of $a\bar{b}c$ and $\bar{a}bc$. In other cases, the relative constraint is labeled as *ow*.

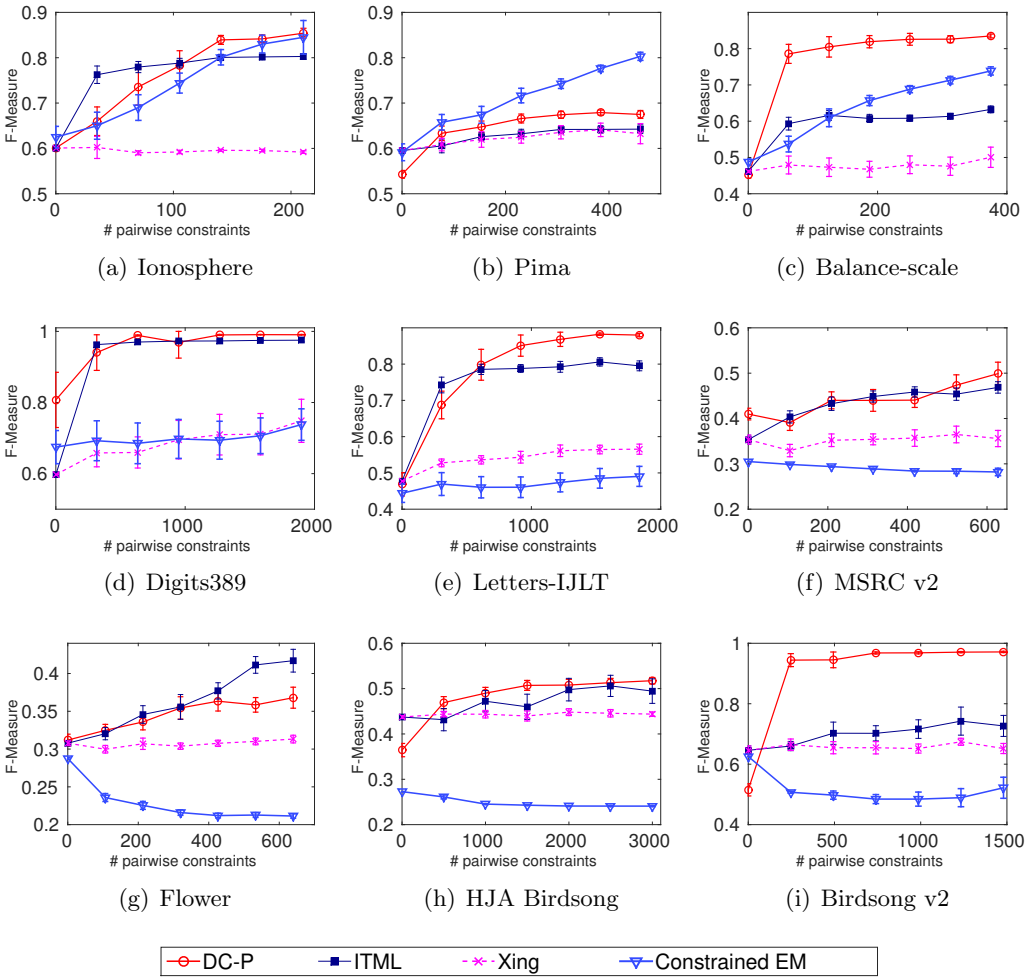
For pairwise constraints, we vary the number of constraints from $0.1N$ to $0.6N$ with a $0.1N$ increment, where N is the total number of instances. We compare our method (shown as DC-P) with Xing, ITML, and Constrained EM using all the pairwise constraints as input for all methods.

For relative constraints, we increase the number of constraints from $0.05N$ to $0.3N$ with a $0.05N$ increment. Since the baseline methods for relative constraints can not easily incorporate the *ow* constraints, we drop the *ow* constraints for these methods. We evaluate our method in two settings, one with all relative constraints as input (shown as *DC-R*), and the other with only $ab\bar{c}, a\bar{b}c, \bar{a}bc$ constraints (shown as *DC-R-C*). That is, DC-R-C uses the same set of constraints with the baseline methods (after dropping the *ow* constraints), while DC-R employs all the generated constraints. The DC-R-C setting is included for two purposes. One is to achieve a fair comparison with the baseline methods as they use the same amount of information from the constraints. The other purpose is to evaluate how the seemingly uninformative *ow* relative constraints can help improving clustering.

We use five-fold cross-validation to tune parameters for all methods using the available constraints, and choose the parameters that maximize the prediction accuracy of on the validation constraints. Our method has three parameters: the coefficient parameters τ and λ, and the constraint relaxation parameter ϵ. We search for the optimal $\tau \in \{0.5, 1, 1.5\}$ and $\lambda \in \{2^{-10}, 2^{-8}, 2^{-6}, 2^{-4}, 2^{-2}\}$. For the choice of ϵ, we empirically observed that our method is very robust to different values when it is within the range $[0.05, 0.15]$. Here we set $\epsilon = 0.05$ for both experiments with pairwise and relative constraints; this allows for soft constraints since the constraints are noise-free. We repeat all experiments for 20 randomized runs each with independently sampled constraint sets, and report the averaged results.

3.3.3.2 Results and Discussions

Figure 3.2 shows the clustering performances with different numbers of *pairwise* constraints for all competing methods. We can see that our method (DC-P) is comparable or outperforms the baselines on all datasets. We would like to note that the proba-



(a) Ionosphere (b) Pima (c) Balance-scale

(d) Digits389 (e) Letters-IJLT (f) MSRC v2

(g) Flower (h) HJA Birdsong (i) Birdsong v2

Figure 3.2: Pairwise Constraints: Clustering performance with different number of constraints. Error bars are reported with mean and the 95% confidence intervals.

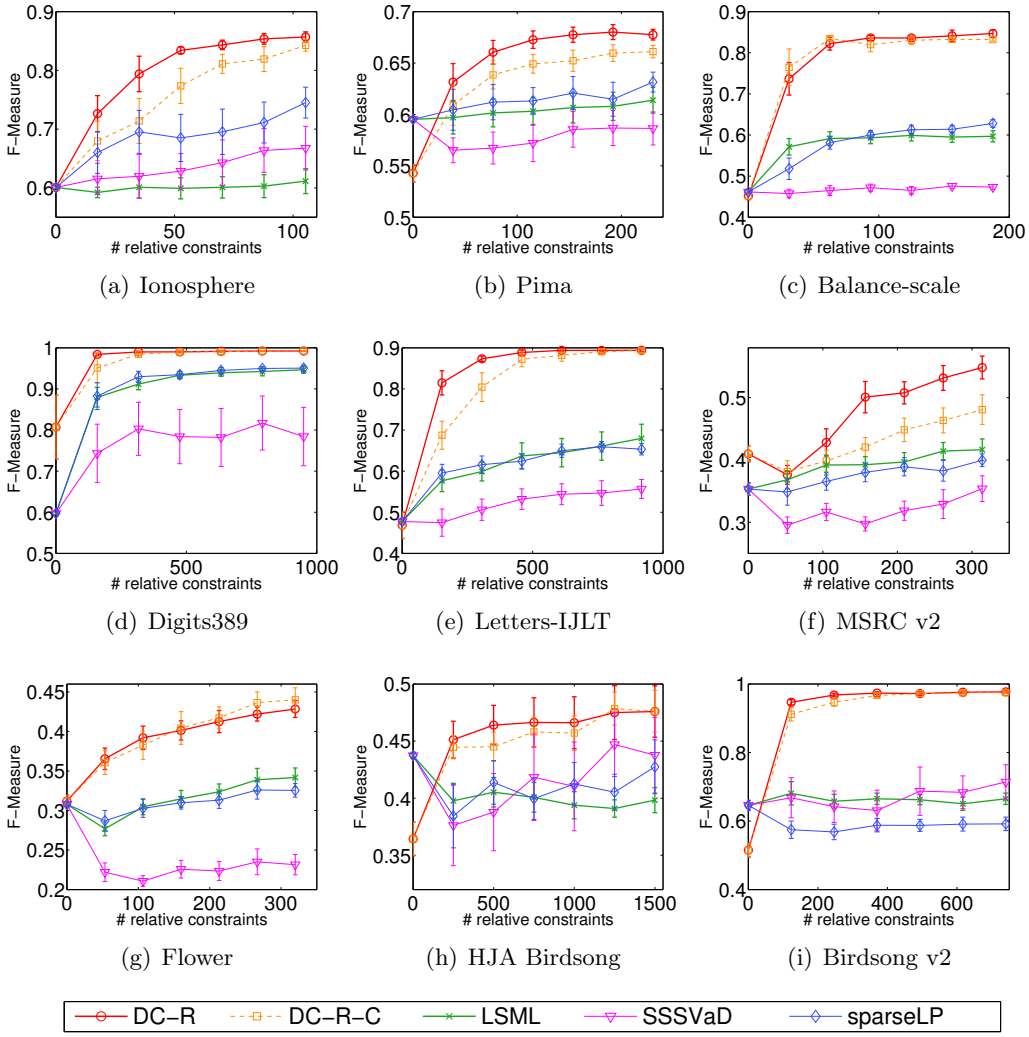


Figure 3.3: Relative Constraints: Clustering performance with different number of constraints. Error bars are reported with mean and the 95% confidence intervals.

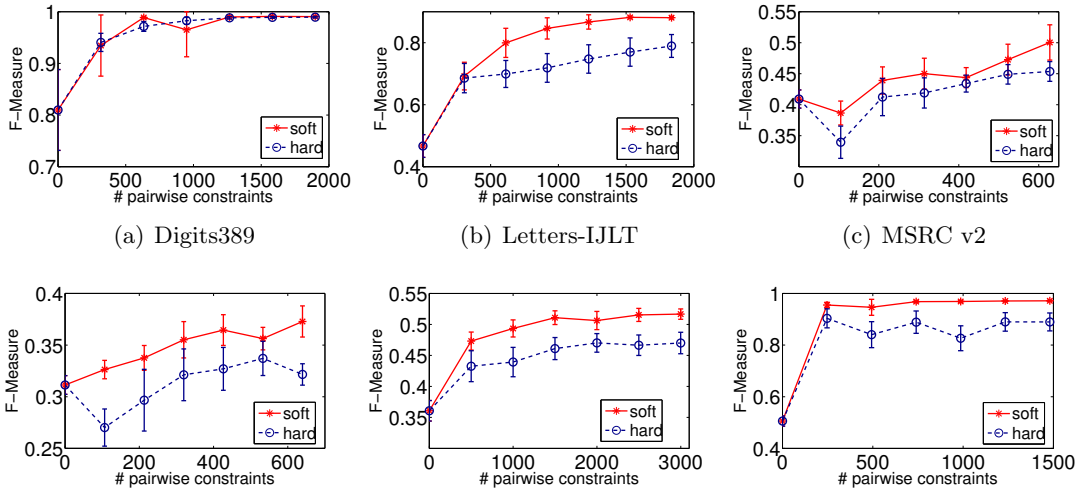
bilistic method Constrained EM only performs reasonably well on datasets with a small number of classes (e.g., *Ionosphere* and *Pima*), but tend to be less effective on datasets with a large number of classes (e.g., *Flower* and *Birdsong v2*). One possible reason is that Constrained EM uses a generative model, which is generally known to be inferior to discriminative models (used in our method) [52, 92] for classification and regression

tasks.

Figure 3.3 shows the performance of different methods with varied number (before dropping *ow* constraints) of *relative* constraints. We see that our method, both DC-R and DC-R-C, consistently outperform all the baselines. The fact that DC-R-C outperforms the baselines suggests that our method is competitive with the state-of-the-art methods, as they are using the same set of constraints. In addition, DC-R often further improves the performance than DC-R-C, which implies that the *ow* constraints are also useful for clustering.

All the above results demonstrate the effectiveness of our unified framework in clustering with either pairwise or relative constraints. This enables us to later compare pairwise and relative constraints using a reasonably good clustering method.

3.3.4 Soft Constraints vs. Hard Constraints

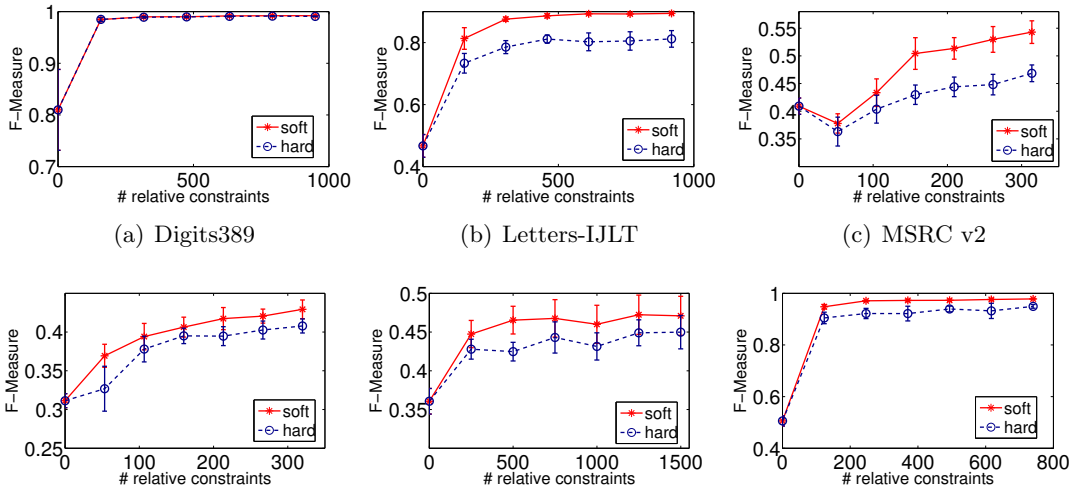


(a) Digits389 (b) Letters-IJLT (c) MSRC v2

(d) Flower (e) HJA Birdsong (f) Birdsong v2

Figure 3.4: Pairwise Constraints: Clustering performance of our method using hard vs. soft constraints.

This set of experiments demonstrates the benefits of allowing for "soft constraints" introduced by the design of conditional probability $P(l|Y)$ in our model. We set $\epsilon = 0.05$ and $\epsilon = 0$ for our method to enforce soft and hard constraints respectively. We compare



(a) Digits389 (b) Letters-IJLT (c) MSRC v2

(d) Flower (e) HJA Birdsong (f) Birdsong v2

Figure 3.5: Relative Constraints: Clustering performance of our method using hard vs. soft constraints.

them by setting the other experiment factors the same with previous section 3.3.3. Figure 3.4 and Figure 3.5 show the results of enforcing soft/hard pairwise/relative constraints on six challenging multi-cluster datasets. The performances of soft/hard constraints on datasets with small number of clusters are similar and are not reported. From the results, we can see that for both cases, using soft constraints generally leads to better performance than using hard constraints. In particular, on the *MSRCv2* and *Flower* datasets, using hard constraints produces a large "dip" at the beginning of the curve while this issue is not as severe for soft constraints (in both cases of pairwise and relative constraints). This suggests that using soft constraints makes our model less susceptible to overfitting to small sets of constraints.

3.3.5 Computational Time

The runtime of learning from pairwise and relative constraints are similar with the same number of constraints. We record the runtime of learning with 1500 relative constraints on our largest HJA Birdsong dataset on a standard desktop computer with 3.4 GHz CPU and 11.6 GB of memory. On average it takes less than 2 minutes to train the model

using an unoptimized Matlab implementation.

3.4 Comparing Informativeness of Constraints: A Mathematical Analysis

The proposed unified framework provides a clustering method that allows for a fair comparison between pairwise and relative constraints. When comparing the two types of constraints, one fundamental question is how to objectively measure their usefulness, regardless of what clustering method is employed. A quite general, sound strategy is to measure their information content. Specifically, we assess the information that each type of constraints provides about the underlying class labels, when the provided constraints are consistent with these class labels. From an information-theoretic perspective, we quantify the information of the constraints by the mutual information between the constraint labels and the clusters labels of the involved instances.

Given i.i.d instances $\{\mathbf{x}_i\}_{i=1}^N$ sampled from K clusters (with replacement), let $p_k = P(y = k)$ be the distribution of the k-th cluster label. Consider a pair $(\mathbf{x}_a, \mathbf{x}_b)$ and a triplet $(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$ with cluster labels $Y^p = [y_a, y_b]^T$ and $Y^r = [y_a, y_b, y_c]^T$, and constraint labels $l^p \in \{\text{ML}, \text{CL}\}$ and $l^r \in \{ab\bar{c}, a\bar{b}c, \bar{a}bc, ow\}$ respectively. As described previously, for noise-free constraints, the labels are determined by (3.1) and (3.2).

The following theorem provides the mutual information between the constraint labels and the associated instance cluster labels.

Theorem 3.4.1. *When constraints are consistent with the class labels as given by (3.1) and (3.2), the mutual information $\mathbb{I}(Y, l)$ for each type of constraints is given by:*

$$\mathbb{I}(Y^p, l^p) = H[l^p] = P_M \log \frac{1}{P_M} + (1 - P_M) \log \frac{1}{1 - P_M}. \tag{3.12}$$

$$\mathbb{I}(Y^r, l^r) = H[l^r] = 3P_R \log \frac{1}{P_R} + (1 - 3P_R) \log \frac{1}{1 - 3P_R}. \tag{3.13}$$

where $P_M = \sum_k p_k^2$ is the marginal probability of obtaining an ML pairwise constraint when choosing two instances uniformly at random, and $P_R = \sum_k p_k^2(1 - p_k)$ is the marginal probability of obtaining an $ab\bar{c}$, $a\bar{b}c$, or $\bar{a}bc$ relative constraint when choosing three instances uniformly at random.

Proof. By the definition of mutual information, $\mathbb{I}(Y, l) = H[l] - H[l|Y]$. Since Y deterministically decides l, we have $H[l|Y] = 0$, and thus $\mathbb{I}(Y, l) = H[l]$. This is true for both pairwise and relative constraints.

Now we derive the specific forms of $H[l^p]$ and $H[l^r]$. For a pairwise constraint, it is easy to see that $P_M = P(l^p = ML) = \sum_k p_k^2$, and $P(l^p = CL) = 1 - P_M$. Hence,

$$H[l^p] = P_M \log \frac{1}{P_M} + (1 - P_M) \log \frac{1}{1 - P_M}.$$

Similarly, for a relative constraint, we have $P_R = P(l^r = ab\bar{c}) = P(l^r = a\bar{b}c) = P(l^r = \bar{a}bc) = \sum_k p_k^2(1 - p_k)$, and $P(l^r = ow) = 1 - 3P_R$. Thus,

$$H[l^r] = 3P_R \log \frac{1}{P_R} + (1 - 3P_R) \log \frac{1}{1 - 3P_R}.$$

This completes our proof. □

Derived from the above theorem, the following corollary compares the two mutual information $\mathbb{I}(Y^p, l^p)$ and $\mathbb{I}(Y^r, l^r)$.

Corollary 3.4.1.1. *If* $p_k \leq \frac{2}{3}, \forall k \in \{1, ..., K\}$ *and* $3P_R \leq 1 - \frac{1}{e}$, *then* $\mathbb{I}(Y^r, l^r) \geq \mathbb{I}(Y^p, l^p)$.

Proof. We show that under the above conditions, both terms of $\mathbb{I}(Y^r, l^r)$ in (3.13) are greater than the corresponding terms of $\mathbb{I}(Y^p, l^p)$ in (3.12).

To prove that $3P_R \log \frac{1}{P_R} > P_M \log \frac{1}{P_M}$ (the first terms), we observe that $P_R = \sum_k p_k^2(1 - p_k) \leq \sum_k p_k^2 = P_M$, from the fact that $1 - p_k \in [0, 1], \forall k$. Then we have

$$\log \frac{1}{P_R} \geq \log \frac{1}{P_M}. \tag{3.14}$$

If $p_k \leq \frac{2}{3}, \forall k$, then $3p_k^2(1 - p_k) \geq p_k^2, \forall k$, and summing over k's we have

$$\sum_k 3p_k^2(1 - p_k) \geq \sum_k p_k^2 \quad \Leftrightarrow \quad 3P_R \geq P_M. \tag{3.15}$$

From the two inequalities derived in (3.14) and (3.15), it is obvious that

$$3P_R \log \frac{1}{P_R} \geq P_M \log \frac{1}{P_M}.$$

To prove that $(1-3P_R)\log\frac{1}{1-3P_R} > (1-P_M)\log\frac{1}{1-P_M}$ (the second terms), we first note that the function $f(x) = (1-x)\log\frac{1}{(1-x)}, 0 \leq x < 1$ is monotonically increasing in the interval $[0, 1-\frac{1}{e}]$. Hence, if $0 \leq P_M \leq 3P_R \leq 1-\frac{1}{e}$ (which holds under the required conditions), then

$$(1-3P_R)\log\frac{1}{1-3P_R} \geq (1-P_M)\log\frac{1}{1-P_M}.$$

Thus, the second term of $\mathbb{I}(Y^r, l^r)$ is also greater than the second term of $\mathbb{I}(Y^p, l^p)$.

Hence, if the two conditions $p_k \leq \frac{2}{3}, \forall k$ and $3P_R \leq 1-\frac{1}{e}$ hold, we have $\mathbb{I}(Y^r, l^r) \geq \mathbb{I}(Y^p, l^p)$. □

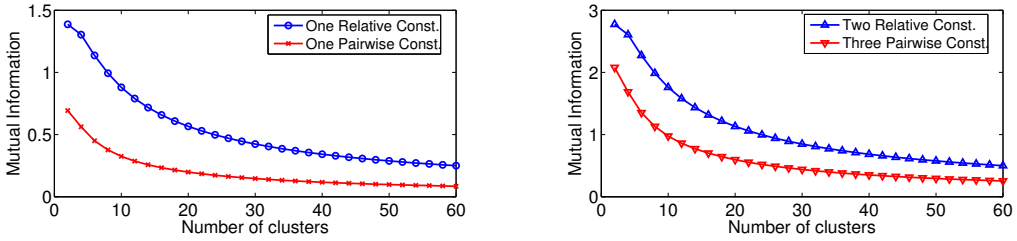
The conditions in Corollary 3.4.1.1 can be interpreted as requirements on the distributions of the cluster and the constraint labels. The first condition $p_k \leq \frac{2}{3}, \forall k$ restricts that none of the clusters contains more than $\frac{2}{3}$ portion of the instances, implying that the clusters are not extremely unbalanced. The second condition indicates that $3P_R \leq 1-\frac{1}{e}$ $P(l^r = ow) = 1-3P_R \geq \frac{1}{e}$, namely the probability of the *ow* relative constraint should not be too small. Combining the two conditions, we can also derive $P_M \leq 3P_R \leq 1-\frac{1}{e}$ and $P(l^p = CL) = 1-P_M \geq \frac{1}{e}$. That is, both the probability of *ow* and CL constraints are larger than $\frac{1}{e}$. Generally, as the number of clusters becomes large, the probability of ow and CL constraints tends to increase. In practice, if there are a large number of clusters that are not extremely imbalanced, then the two conditions are easily satisfied and we have $\mathbb{I}(Y^r, l^r) \geq \mathbb{I}(Y^p, l^p)$.

We note that the conditions in Corollary 3.4.1.1 are sufficient but not necessary. In some cases where such conditions are not satisfied, we may still have $\mathbb{I}(Y^r, l^r) \geq \mathbb{I}(Y^p, l^p)$. For example, when the cluster labels are equally distributed, i.e., $p_k = \frac{1}{K}, \forall k$, the condition $3P_R \leq 1-\frac{1}{e}$ is not always satisfied for different values of K. However, below we show that $\mathbb{I}(Y^r, l^r) > \mathbb{I}(Y^p, l^p)$ still holds.

In this case, we substitute the values of p_k's to (3.12) and (3.13) and obtain

$$\mathbb{I}(Y^p, l^p) = \log K - \left(1-\frac{1}{K}\right)\log(K-1)$$

$$\mathbb{I}(Y^r, l^r) = 2\log K - \frac{3(K-1)}{K^2}\log(K-1) - \left(1-\frac{3(K-1)}{K^2}\right)\log[K^2-3(K-1)].$$



(a) Same number of *constraints* (b) Same number of *instances*

Figure 3.6: Comparison of the mutual information between the pairwise and relative constraint labels and the instance cluster labels as a function of the number of clusters.

To compare $\mathbb{I}(Y^p, l^p)$ and $\mathbb{I}(Y^r, l^r)$, we plot their values as a function of the number of clusters K in Figure 3.6. Comparing the values of *one relative constraint* and *one pairwise constraint* in Figure 3.6(a), we see that $\mathbb{I}(Y^r, l^r) > \mathbb{I}(Y^p, l^p)$ for different values of K's. One might argue that a triplet contains more instances than a pair, which may make this comparison unfair. To address this potential bias, we compare them in the case where the same number of instances are involved in both types of constraints, i.e., two relative and three pairwise constraints. For simplicity we consider the case of disjoint constraints, where the mutual information of two relative and three pairwise constraints are simply the corresponding multiple of one constraint. Figure 3.6(b) plots the information of *two relative constraints* and *three pairwise constraints*, both involving six instances. We again see that relative constraints are more informative.

Overall, the above analyses strongly suggest that relative constraint is a more effective representation of domain knowledge for clustering.

3.5 Empirical Comparison of Constraints via A User Study

In this section, we empirically compare the two types of constraints through a user study and answer the three research questions proposed in Section 3.1.

3.5.1 User Study Design

3.5.1.1 Participants and Tasks

We recruit 24 participants from a pool of responses to a campus-wide recruitment notice. None of the participants has any experience with the specifics of clustering with constraints or with the datasets used in the study.

Two treatments are established in the study: labeling 195 pairwise constraints and labeling 130 relative constraints. We employ datasets from two domains: 1) the Birdsong v2 dataset associated with a bioacoustics application, where the goal is to find distinct birdsong patterns by detecting clusters of birdsong segments and 2) the Flower dataset from a more general image domain, where the goal is to identify the species of each flower in the image.

We adopt a within-subject design where each participant works with every treatment and dataset (i.e., each participant labels both pairwise and relative constraints for both datasets). Within each subject, the tasks are performed in random order. For each task, we first sample each constraint set from a large pool of randomly generated constraints, and then post queries to the users.

3.5.1.2 Procedures

In the pre-task introduction, we explain the concepts of clustering with pairwise and relative constraints, the main tasks and the user interface. We also give the participants a few practice examples. During the experiments, we ask the participants to label the constraints by sequentially showing the images of the compared items. Due to their different backgrounds, participants may have different familiarity levels with the datasets. For example, engineering students might be more familiar with the spectrogram images in the Birdsong v2 datasets than others. To reduce such potential bias, we provide a reference book for each dataset listing one example from each of the classes. The reference book is explained to the participants before the tasks and is also available during the tasks. We record the time it takes for each participant to respond each query (i.e., to provide a constraint).

(a) Pairwise Constraint (b) Relative Constraint

Figure 3.7: Screenshots of obtaining pairwise and relative constraints for the Flower dataset using the developed interface.

3.5.1.3 User Interface

We design a web-based interface to acquire both types of constraints by posing queries to the participants. Specifically, for each participant, we first sample the constraint sets from a large pool manually labeled with "easy" or "difficult" constraints and fix the difficulty level for different participants by controlling the number of "easy" and "difficult" constraints in each set. We then query the constraints to the participants sequentially in random ordering. For pairwise constraints, we display image pairs to the participant and ask, "*Do these images belong to the same class?*". Similarly, for relative constraints, we present the image triplets and ask the participants "*Select two images that you think belong to the same class, which is different from the class of the other image. Click 'otherwise' if you don't think such pair exists.*". Figure 3.7 shows an example for each type of constraints from the Flower dataset with our interface.

3.5.1.4 Post-Task Questionnaires

After finishing all the tasks, we ask the participants to take a survey regarding their labeling experiences. The survey contains the following questions, some based on the NASA-TLX questionnaire [47] and others designed specifically for the study:

- `PERFORMANCE` Regarding both datasets, select the task where you have more confidence about your answers.
- `EFFORT PER QUERY` On average, select the task where you spent more effort to

Table 3.6: The average time cost (mean ± std in seconds) for labeling each constraint.

(a) Pairwise Constraints

Dataset	Per Const	Per ML	Per CL
Birdsong	3.08 ± 1.22	3.95 ± 1.68	2.84 ± 1.28
Flower	2.10 ± 0.71	2.41 ± 0.71	2.04 ± 0.75

(b) Relative Constraints

Dataset	Per Const	$ab\bar{c}/a\bar{b}c/\bar{a}bc$	Per ow
Birdsong	6.19 ± 3.17	6.44 ± 3.21	5.98 ± 3.25
Flower	4.05 ± 1.41	4.02 ± 1.23	4.09 ± 1.63

complete each query on the two datasets.

- `OVERALL EFFORT` On average, select the task where you spent more effort to complete all pairwise queries (195) and relative queries (130) on the two datasets.
- `FRUSTRATION` Regarding both datasets, select the task where you felt more stressed or discouraged.
- `PREFERENCE` Based on your experience of labeling pairwise and relative constraints on both datasets, which task do you prefer in general?

3.5.2 Results and Discussions

In this section, we evaluate the user study results to compare the constraints' impacts on the human users and the clustering systems.

3.5.2.1 Ease of Obtaining Constraints

We investigate two aspects to compare the ease of obtaining the two types of constraints (Q2). First, we hypothesize that the cost of obtaining the constraints is related to the time spent on labeling the constraints. Then we compare the labeling time for each type of constraints. Second, we compare the participants' own preferences based on their labeling experiences.

Table 3.6 reports the average time for labeling each pairwise, ML, and CL constraint,

Table 3.7: Survey results regarding preferences between labeling pairwise and relative constraints.

Factors	Pairwise	Relative	No preference
Better Performance	16/24	6/24	2/24
Less Effort per Query	17/24	5/24	2/24
Less Effort for all Queries	15/24	6/24	3/24
Less Frustration	13/24	3/24	8/24
Overall Preference	17/24	4/24	3/24

and the time for labeling each relative constraint, $ab\bar{c}, a\bar{b}c, \bar{a}bc$ constraint, and *ow* constraint. First, we see that for both datasets labeling a pairwise constraint requires less amount of time than labeling a relative constraint. One possible reason could be that each relative constraint involves more instances to be examined and more pairwise comparisons between instances to be made, which may result in longer labeling time. This result implies that pairwise constraints might be easier to obtain if the labeling effort is based on the time spent. Second, we also observe that the time of labeling a CL constraint is shorter than that of ML constraints, implying that CL constraints are easier to label than ML constraints.

Table 3.7 reports the statistics of participants' answers to the survey questions. We see that most participants feel more confident on their labels to pairwise constraints and also spend less effort on the labeling. This suggests that from the users' perspective, providing pairwise constraints is in general easier.

Overall, 18 participants prefer to provide supervision in the form of pairwise constraints for similar reasons. For example, some comments from the participants are,

> *"It is easier to compare two different items than comparing three items at a time."*

> *"It is easier for me to compare two things and quickly judge whether they share a quantity of characteristics that place them in the same class. For many pairwise comparisons, I could quickly see that they fit vastly in different classes."*

These answers suggest that Cannot-link (CL) pairwise constraints are in general easier for the user to provide than relative constraints. In the queried constraint sets, there is

usually a large portion of CL constraints (due to the random sampling from the dataset of a large number of classes), which may explain why the participants feel easier to provide pairwise constraints. However, four users prefer to use relative constraints. Some of the reasons are

> *"It seems to be something innate, I am reminded of Sesame Street, 'one of these things is different, of these things does not belong.' Choosing the odd one out of three is easier than are these the same."*

> *"I think that it is easier to distinguish patterns differing in a group. There can be variations within the same class but with two different classes, the patterns are easier to distinguish especially when there are three subjects to compare."*

From these comments, we hypothesize that it is easier to pick out one different instance for the relative constraint query than judging the similarities of a pair when the pairwise constraint is ML. That is, an additional instance in the relative constraint provides more context information about the patterns, which makes the $ab\bar{c}, a\bar{b}c, \bar{a}bc$ relative constraints easier to label than ML pairwise constraints.

Some participants have no preference between providing pairwise or relative constraints. For example, one user said that

> *"Pairwise constraints seem easier to do. Relative constraints take more time. However, I will say that in the more difficult Birdsong task, the relative constraints were helpful because it gave me something other than the reference book to compare to."*

In summary, CL pairwise constraints seem easier to obtain than relative constraints. Likewise, the $ab\bar{c}, a\bar{b}c, \bar{a}bc$ relative constraints tend to be easier to acquire than ML pairwise constraints. Looking back at the sampled constraint sets used in this experiment, we noted that the CL pairwise constraints made up the larger proportion of pairwise constraints for each user due to that random sampling would produce more CL constraints than ML constraints as the number of classes increases. This is likely to have influenced the perception that pairwise constraints are in general easier to label than relative constraints.

Table 3.8: Birdsong v2 Data: The average confusion matrix of the human labeled constraints vs. the ground truth constraint labels. The overall accuracy for each type of constraint is shown in the corresponding caption.

(a) Pairwise Const. (83.52%)

True	Human Labels	
	CL	ML
CL	142.42	6.50
ML	8.92	37.15

(b) Relative Const. (82.68%)

True	Human Labels			
	ow	$ab\bar{c}$	$a\bar{b}c$	$\bar{a}bc$
ow	59.88	3.46	3.38	3.62
$ab\bar{c}$	2.62	14.73	0.12	0.62
$a\bar{b}c$	3.65	0.31	17.00	0.38
$\bar{a}bc$	3.62	0.31	0.42	15.88

Table 3.9: Flower Data: The average confusion matrix of the human labeled constraints vs. the ground truth constraint labels. The overall accuracy for each type of constraint is shown in the corresponding caption.

(a) Pairwise Const. (96.65%)

True	Human Labels	
	CL	ML
CL	158.54	4.54
ML	2.00	29.92

(b) Relative Const. (92.88%)

True	Human Labels			
	ow	$ab\bar{c}$	$a\bar{b}c$	$\bar{a}bc$
ow	75.88	1.46	1.73	2.15
$ab\bar{c}$	0.88	14.12	0.08	0.15
$a\bar{b}c$	1.19	0.12	13.42	0.15
$\bar{a}bc$	1.23	0.08	0.04	17.31

3.5.2.2 Labeling Accuracy

Next, we evaluate the labeling accuracy of the two types of constraints and address the research question Q3 in section 3.1. We measure the accuracy of the constraint labels against the labels derived from the ground truth instance classes. Table 3.8 and Table 3.9 list the average confusion matrix of the human-labeled constraints against the ground truth.

From the results, we see that the overall accuracy for labeling pairwise constraints on both datasets is slightly higher than that of relative constraints. Several possible reasons may explain the results. One is that the CL pairwise constraints are very easy to distinguish and the large portion of CL constraint may lead to overall higher accuracy for pairwise constraints. Another possible reason could be that the multiple options for labeling relative constraints make the chance of mistakes naturally higher than that of

pairwise constraints (where there are only two choices), resulting in overall higher error rate.

However, one interesting result is that the confusion of $ab\bar{c}, a\bar{b}c, \bar{a}bc$ for relative constraints is much smaller than that of ML and CL constraints. This implies that users are very accurate at providing relative constraints whenever two of the compared instances belong to the same class (while the third instance does not).

This user study result also points to possible ways to further improve our model. As revealed by Table 3.8 and 3.9, the noise on the labels for relative constraints is not uniform as assumed by our model. An interesting future direction is to introduce a non-uniform noise process to more realistically model the users' labeling behaviors.

3.5.2.3 Effectiveness in Improving Clustering

Now we compare the effectiveness of the two types of constraints and answer Q1 in Section 3.1. We consider the following two factors:

- **Given a limited budget for examining instances, which type of constraint will be more efficient for clustering?**

  We control the number of pairwise constraints to be 1.5 times the number of relative constraints and evaluated clustering performance using both synthetic and human-labeled constraints. For synthetic constraints, we randomly generate relative constraints with size 0.05N to 0.3N with a 0.05N increment, and the competing pairwise constraints with size 0.075N to 0.45N with a 0.075N increment. For human-labeled constraints, We divide the constraints set obtained for each user into five sets of equal size and incrementally add each set. We run clustering with such constraints on all datasets used in Section 3.3.

- **Given limited time for labeling the constraints, which type of constraint can improve clustering performance more significantly?**

  We form the comparison sets using the pairwise and relative constraints obtained within the same amount of time in the user study. In particular, let T_i^p and T_i^r be the total time that the i-th participant spent on labeling all pairwise and relative constraints respectively. We form the incrementally add constraints obtained within time $0.2T_i$, $0.4T_i$ up to T_i.

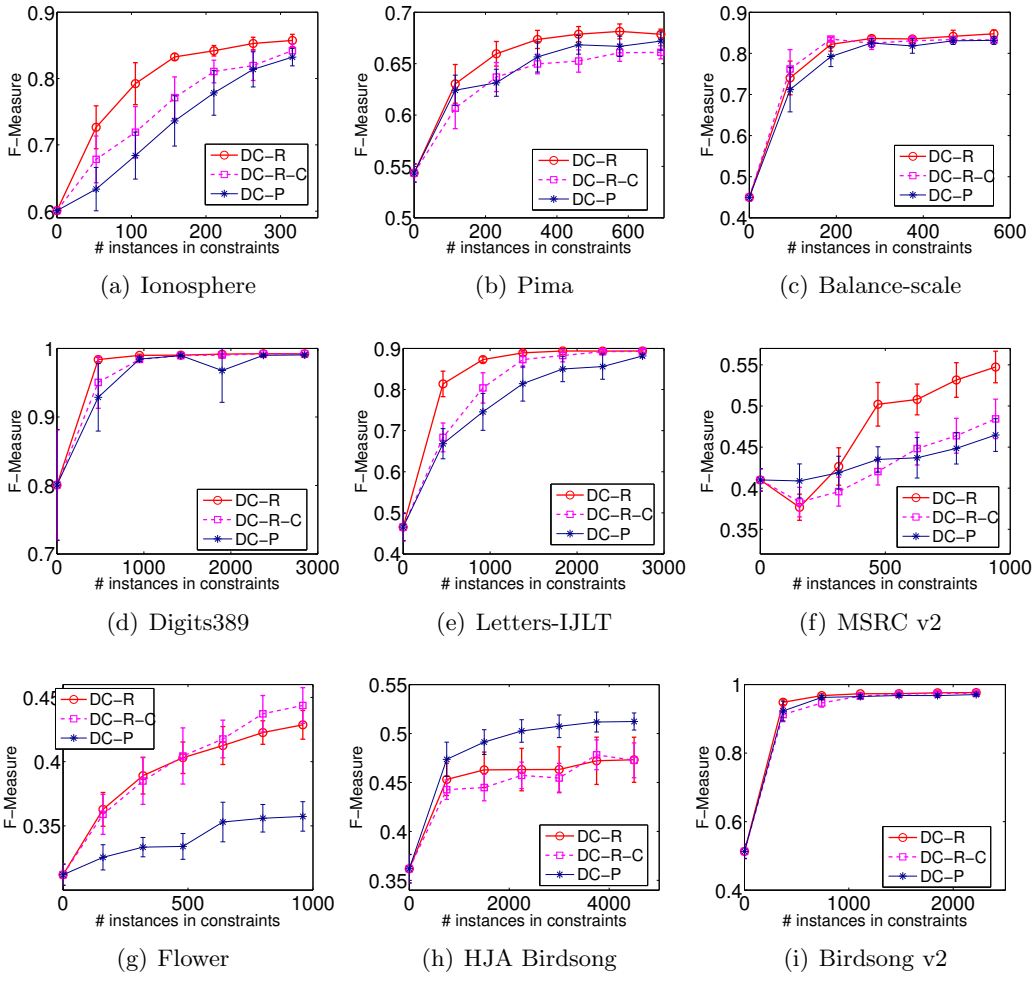
As stated previously, to ensure a fair comparison, we use our unified framework to clustering with the two types of constraints. The parameter ϵ is set to 0.05 for the synthetic noise-free constraints again. For the human-labeled constraints, we set $\epsilon = 0.15$ to account for labeling noise[4].

Figure 3.8 shows the results of clustering using synthetic constraints containing the *same number of instances* (Here DC-R uses relative constraints containing the same number of instances with pairwise constraints employed by DC-P, while DC-R-C drops the *ow* constraints used by DC-R). We can observe that on most datasets, the clustering performance with relative constraints (both DC-R and DC-R-C) is higher than that with pairwise constraints (DC-P). One exception is the HJA Birdsong dataset, where DC-P performs much better than DC-R and DC-R-C. One possible reason is that the extremely unbalanced cluster distribution in the data causes the pairwise constraints to be more informative than relative constraints, as analyzed in Corollary 3.4.1.1. This in turn results in a better clustering performance.

Figure 3.9 reports the clustering performance using the human-labeled constraints regarding both factors mentioned above. We can see that using relative constraints leads to higher performance improvements than using pairwise constraints in both cases. Figure 3.9(c) shows that when the labeling time is very limited, pairwise constraints provide more improvement on the clustering results on the less difficult Birdsong v2 dataset. But it is not the case for the more difficult Flower dataset. We suspect that more constraints are needed to significantly improve the clustering on the Flower datasets due to its difficulty. Another observation is that using only $ab\bar{c}, a\bar{b}c, \bar{a}bc$ constraints on the Flower dataset gives better performance compared to using the additional *ow* constraints. This implies that on difficult tasks, it is more useful to use constraints that provide explicit instance cluster membership information.

To further analyze the results in Figure 3.9, we examine the number of constraints used in the human-labeled constraints. Figure 3.10 reports the number of all constraints and the number of correctly labeled constraints. It is interesting to see that for both settings, we used far more pairwise constraints than relative constraints on both datasets. In addition, from prior results, the noise in pairwise constraints is smaller than that in relative constraints. The fact that relative constraints still provide more improvement

[4]For these noisy constraints, our method is robust to the choice of ϵ. Using different values of $\epsilon \in [0.05, 0.2]$ only results in minor fluctuations to the F-measure.

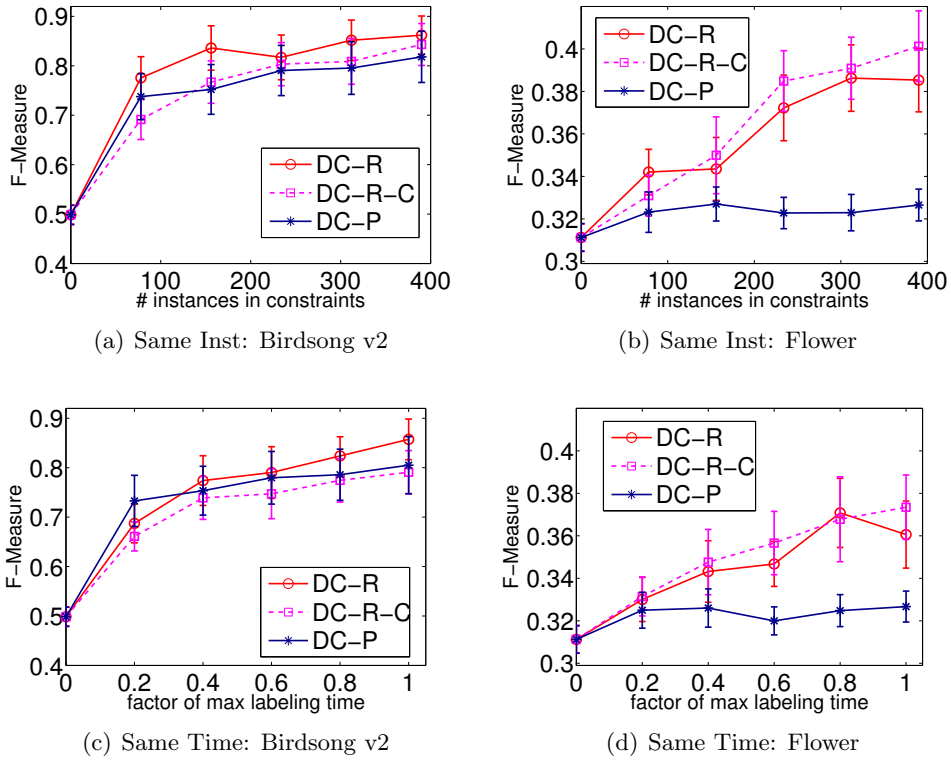


(a) Ionosphere (b) Pima (c) Balance-scale

(d) Digits389 (e) Letters-IJLT (f) MSRC v2

(g) Flower (h) HJA Birdsong (i) Birdsong v2

Figure 3.8: Clustering performance with synthetic pairwise and relative constraints with the same number of instances. Results are averaged over 20 runs. Error bars show the 95% confidence intervals.



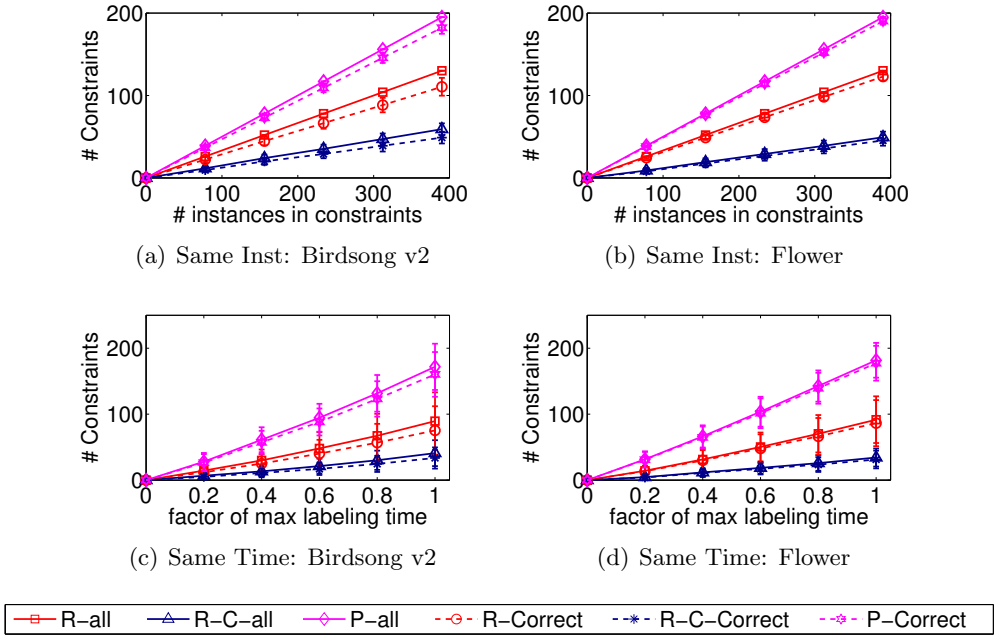
(a) Same Inst: Birdsong v2

(b) Same Inst: Flower

(c) Same Time: Birdsong v2

(d) Same Time: Flower

Figure 3.9: Clustering performance with human-labeled pairwise and relative constraints. "Same Inst": two types of constraints involve the same number of instances. "Same Time": two types of constraints are obtained within the same amount of time. Results are averaged over 24 users. Error bars show the 95% confidence intervals.



(a) Same Inst: Birdsong v2

(b) Same Inst: Flower

(c) Same Time: Birdsong v2

(d) Same Time: Flower

Figure 3.10: The number of all labeled constraints and correctly labeled constraints used in the comparison of Figure 3.9. "R": relative constraints; "R-C": $ab\bar{c}, a\bar{b}c, \bar{a}bc$ relative constraints; "P": pairwise constraints; "-all": total number of constraints;"-Correct": correctly labeled constraints.

further demonstrates that relative constraint is more effective.

Overall, our results reveal that labeling relative constraints might require higher cost than labeling pairwise constraints. However, they may still be more useful since, from our results, they are more effective at improving clustering than pairwise constraints obtained with the same amount of or even more labeling effort. We believe that our results can be generalized to other clustering methods that can learn from both types of constraints labeled in the considered setting.

3.6 Related Work

Pairwise Constraints: Many studies exist on the topic of clustering with pairwise constraints. These approaches mainly fall into two categories. The first category incorporate the constraints to clustering by relating them to instance clusters. Namely, pairs of instances in ML constraints should be grouped to the same cluster while those in CL constraints should be divided into different clusters. Such information is then used to learn a clustering model that respect the constraints [7, 36, 53, 59, 60, 68, 70, 71, 76, 90, 102, 106, 107, 121, 123]. Differently, the second category uses the constraints in a more moderate way by deriving distance relations between instances. Methods in this category usually enforce that instances linked with ML constraints should have short distances between them, while those connected with CL constraint should be separated by large distances [4, 22, 31, 74, 109, 110, 117]. By imposing such constraints, a distance metric is learned and later used for clustering. There are also methods that combine these two types of techniques [13], and find the clustering solution along with a distance metric during the learning.

Relative Constraints: The study of relative constraints is relatively less extensive than that of pairwise constraints. There are also several ways to learn from such constraints. One common approach is to impose distance relations [2, 23, 49, 58, 66, 83, 89]. Such methods enforce $d(\mathbf{x}_a, \mathbf{x}_b) < d(\mathbf{x}_a, \mathbf{x}_c)$ for the constraint that $\mathbf{x}_a$ is more similar to $\mathbf{x}_b$ than to $\mathbf{x}_c$, where $d(\cdot)$ is the distance function. Another type of approach directly links the similarity comparison to instance cluster memberships. The work [80] relates the cluster membership to three possible constraint labels, "yes", "no", and "dnk", representing three possible comparison outcomes. The "yes" ("no") label means that $\mathbf{x}_a$ is more similar to $\mathbf{x}_b$ ($\mathbf{x}_c$) than to $\mathbf{x}_c$ ($\mathbf{x}_b$). The "dnk" label includes other cases that are not

covered by the "yes" and "no". The authors use a probabilistic model to learn from such constraints. In addition, another type of methods represents the instance relative similarities in the form of hierarchical ordering [3, 67] and finds hierarchical clustering structure in the data.

Comparison Between Pairwise and Relative Constraints: Although many work exist regarding both types of constraints, very few study has focused on comparing the two. Earlier work on relative constraints [58, 80, 83] compared their proposed methods with existing methods for pairwise constraints. However, the experiments were not designed to make fair comparisons. Such comparisons usually use different types of algorithms for different types of constraints. As a result, it is not clear whether the performance difference comes from the use of different types of constraints or the clustering algorithms. In contrast, this chapter proposed a unified framework for clustering, which made a fair comparison possible. Moreover, this chapter not only investigated the efficiency of the two types of constraint in improving clustering, but also studied the cost of acquiring both types of constraints as well as their labeling accuracy.

3.7 Conclusions

In this chapter, we presented an extensive comparison between *pairwise constraints* and *relative constraints*, two common types of constraints employed to aid clustering. Specifically, we studied three research questions: from the point of view of obtaining constraints, we studied which type of constraint is easier to obtain from the human labelers and which type of constraint allows for higher labeling accuracy. From another point of view of applying constraints to aid clustering, we compared the effectiveness of the two types of constraints in improving clustering. In addition, to ensure the comparison of clustering is performed on equal ground, we also proposed an effective unified framework that can incorporate either type of constraints to clustering. Our results reveal that pairwise constraints are generally easier to obtain, especially when a large portion of CL constraints exist. However, using the unified clustering model, the relative constraints are more effective, even if the number of used relative constraints is far less than that of pairwise constraints.

Manuscript 3: Bayesian Active Clustering with Pairwise Constraints

Yuanli Pei, Li-Ping Liu and Xiaoli Z. Fern

Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)
Edited by Appice A., Rodrigues P., Santos Costa V., Soares C., Gama J., Jorge A.
Published by Springer, 2015
Pages 235–250

Chapter 4: Bayesian Active Clustering with Pairwise Constraints

Abstract

Clustering can be improved with *pairwise constraints* that specify similarities between pairs of instances. However, randomly selecting constraints could lead to the waste of labeling effort, or even degrade the clustering performance. Consequently, how to *actively* select effective pairwise constraints to improve clustering becomes an important problem, which is the focus of this chapter. In this work, we introduce a Bayesian clustering model that learns from pairwise constraints. With this model, we present an active learning framework that iteratively selects the most informative pair of instances to query an oracle, and updates the model posterior based on the obtained pairwise constraints. We introduce two information-theoretic criteria for selecting informative pairs. One selects the pair with the most uncertainty, and the other chooses the pair that maximizes the marginal information gain about the clustering. Experiments on benchmark datasets demonstrate the effectiveness of the proposed method over state-of-the-art.

4.1 Introduction

Constraint-based clustering aims to improve clustering using user-provided *pairwise constraints* regarding similarities between pairs of instances. In particular, a must-link constraint states that a pair of instances belong to the same cluster, and a cannot-link constraint implies that two instances are in different clusters. Existing work has shown that such constraints can be effective at improving clustering in many cases [4, 7, 32, 70, 76, 90, 102, 110, 121]. However, most prior work focus on "passive" learning from constraints, i.e., instance pairs are randomly selected to be labeled by a user. Constraints acquired in this random manner may be redundant and lead to the waste of labeling effort, which is typically limited in real applications. Moreover, when the constraints are not properly selected, they may even be harmful to the clustering performance as has been revealed by Davidson et al. [30]. In this chapter, we study the important problem of *actively* selecting effective pairwise constraints for clustering.

Existing work on active learning of pairwise constraints for clustering has mostly focused on neighbourhood-based methods [6, 46, 50, 72, 111]. Such methods maintain a neighbourhood structure of the data based on the existing constraints, which represents a partial clustering solution, and they query pairwise constraints to expand such neighborhoods. Other methods that do not rely on such structure consider various criteria for measuring the utility of instance pairs. For example, Xu et al. [112] propose to select constraints by examining the spectral eigenvectors of the similarity matrix, and identify data points that are at or close to cluster boundaries. Vu et al. [101] introduce a method that chooses instance pairs involving points on the sparse regions of a k-nearest neighbours graph. As mentioned by Xiong et al. [111], many existing methods often select a batch of pairwise constraints before performing clustering, and they are not designed for iteratively improving clustering by querying new pairs.

In this work, we study Bayesian active clustering with pairwise constraints in an iterative fashion. In particular, we introduce a Bayesian clustering model to find the clustering posterior given a set of pairwise constraints. At every iteration, our task is: a) to select the most informative pair toward improving current clustering, and b) to update the clustering posterior after the query is answered by an oracle/a user. Our goal is to achieve the best possible clustering performance with minimum number of queries.

In our Bayesian clustering model, we use a discriminative logistic model to capture the conditional probability of the cluster assignments given the instances. The likelihood of observed pairwise constraints is computed by marginalizing over all possible cluster assignments using message passing. We adopt a special data-dependent prior that encourages large cluster separations. At every iteration, the clustering posterior is represented by a set of samples ("particles"). After obtaining a new constraint, the posterior is effectively updated with a sequential Markov Chain Monte Carlo (MCMC) method ("particle filter").

We present two information-theoretic criteria for selecting instance pairs to query at each iteration: a) *Uncertain*, which chooses the most uncertain pair based on current posterior, and b) *Info*, which selects the pair that maximizes the information gain regarding current clustering. With the clustering posterior maintained at every iteration, both objectives can be efficiently calculated.

We evaluate our method on benchmark datasets, and the results demonstrate that our Bayesian clustering model is very effective at learning from a small number of pairwise

constraints, and our active clustering model outperforms state-of-the-art active clustering methods.

4.2 Problem Statement

The goal of clustering is to find the underlying cluster structure in a dataset $X = [x_1, \cdots, x_N]$ with $x_i \in \mathbb{R}^d$ where d is the feature dimension. The unknown cluster label vector $Y = [y_1, \cdots, y_N]$, with $y_i \in \{1, \cdots, K\}$ being the cluster label for x_i, denotes the ideal clustering of the dataset, where K is the number of clusters. In the studied *active* clustering, we could acquire some weak supervision, i.e., pairwise constraints, by requesting an oracle to specify whether two instances $(x_a, x_b) \in X \times X$ belong to the same cluster. We represent the response of the oracle as a pair label $z_{a,b} \in \{+1, -1\}$, with $z_{a,b} = +1$ representing that instance x_a and x_b are in the same cluster (a must-link constraint), and $z_{a,b} = -1$ meaning that they are in different clusters (a cannot-link constraint). We assume the cost is uniform for different queries, and the goal of active clustering is to achieve the best possible clustering with the least number of queries.

In this work, we consider sequential active clustering. In each iteration, we select one instance pair to query the oracle. After getting the answer of the query, we update the clustering model to integrate the supervision. With the updated model, we then choose the best possible pair for the next query. So the task of active clustering is an iterative process of posing queries and incorporating new information to clustering.

An active clustering model generally has two key components: the *clustering* component and the *pair selection* component. In every iteration, the task of the clustering component is to identify the cluster structure of the data given the existing constraints. The task of the pair selection component is to score each candidate pair and choose the most informative pair to improve the clustering.

4.3 Bayesian Active Clustering

4.3.1 The Bayesian Clustering Model

In our model, we assume that the instance cluster labels y_i's are independent given instance x_i and the model parameter W. Each pair label $z_{a,b}$ only depends on the cluster

labels y_a and y_b of the involved instances (x_a, x_b). The proposed Bayesian clustering model consists of three elements: 1) the instance cluster assignment model defined by $P(Y|W, X)$, with parameter W; 2) the conditional distribution of the pair labels given the cluster labels $P(Z|Y)$, where Z contains all pair labels in the constraints; and 3) the data-dependent prior $P(W|X, \theta)$ with parameter θ. The joint distribution of the clustering model is factorized as

$$P(Z, Y, W|X, \theta) = P(Z|Y)P(Y|W, X)P(W|X, \theta) \ . \tag{4.1}$$

We use the following discriminative logistic model as the clustering assignment model $P(Y|W, X)$:

$$P(y_i = k|W, x_i) = \frac{\exp(W_{\cdot,k}^\top x_i)}{\sum_{k'=1}^{K} \exp(W_{\cdot,k'}^\top x_i)}, \quad \forall 1 \leq k \leq K, \quad 1 \leq i \leq N \ , \tag{4.2}$$

where W is a $d \times K$ matrix, d is the feature dimension, and K is the number of clusters.

Here we use a special prior for W, which combines the Gaussian prior with a data-dependent term that encourages large cluster separations of the data. The logarithmic form of the prior distribution is

$$\log \ P(W|X, \theta) = -\frac{\lambda}{2}\|W\|_F^2 - \frac{\tau}{N}\sum_{i=1}^{N} H(y_i|W, x_i) + constant \ , \tag{4.3}$$

where the prior parameter $\theta = [\lambda, \tau]$. The first term is the weighted Frobenius norm of W. This term corresponds to the Gaussian prior with zero mean and diagonal covariance matrix with λ as the diagonal elements, and it controls the model complexity. The second term is the average negative entropy of the cluster assignment variable Y. We use this term to encourage large separations among clusters, as similarly utilized by [45] for semi-supervised classification problems. The constant term normalizes the probability. Although it is unknown, inference can be carried out by sampling from the unnormalized distribution (e.g., using slice sampling [75]). We will discuss more details in Sec. 4.3.3.

With our model assumption, the conditional probability $P(Z|Y)$ is fully factorized based on the pairwise constraints. For a single pair (x_a, x_b), we define the probability of

$z_{a,b}$ given cluster labels y_a and y_b as

$$\begin{aligned} P(z_{a,b} = +1|y_a, y_b) = & \begin{cases} \epsilon & \text{if } y_a \neq y_b \\ 1-\epsilon & \text{if } y_a = y_b \end{cases} , \\ P(z_{a,b} = -1|y_a, y_b) = & \ 1 - P(z_{a,b} = +1|y_a, y_b) \ , \end{aligned} \tag{4.4}$$

where ϵ is a small number to accommodate the (possible) labeling error. In the case where no labeling error exists, ϵ allows for "soft constraints", meaning that the model can make small errors on some pair labels and achieve large cluster separations.

4.3.1.1 Marginalization of Cluster Labels.

In the learning procedure described later, we will need to marginalize some or all cluster labels, for example, in the case of computing the likelihood of the observed pair labels:

$$P(Z|W, X) = \sum_{Y} P(Z, Y|W, X) = \sum_{Y_{\alpha(Z)}} P(Z|Y_{\alpha(Z)}) P(Y_{\alpha(Z)}|W, X_{\alpha(Z)}) \ , \tag{4.5}$$

where $\alpha(Z)$ denotes the set of indices for all instances involved in Z.

The marginalization can be solved by performing sum-product message passing [56] on a factor graph defined by all the constraints. Specifically, the set of all instances indexed by $\alpha(Z)$ defines the nodes of the graph, and $P(Y_{\alpha(Z)}|W, X_{\alpha(Z)})$ defines the node potentials. Each queried pair (x_a, x_b) creates an edge, and the edge potential is defined by $P(z_{a,b}|y_a, y_b)$. In this work, we require that the graph formed by the constraints does not contain cycles, and message passing is performed on a tree (or a forest, which is a collection of trees). Since inference on trees are exact, the marginalization is computed exactly. Moreover, due to the simple form of the edge potential (which is a simple modification to the identity matrix as can be seen from (4.4)), the message passing can be performed very efficiently. In fact, each message propagation only requires $O(K)$ complexity instead of $O(K^2)$ as in the general case. Overall the message passing only takes $O(K|Z|)$, even faster than calculating the node potentials $P(Y_{\alpha(Z)}|W, X_{\alpha(Z)})$, which takes $O(dK|Z|)$.

4.3.2 Active Query Selection

Now we describe our approach for actively selecting informative pairs at every iteration. Suppose our query budget is T. In each iteration $t, 1 \leq t \leq T$, we need to select a pair $(x_a^t,\ x_b^t)$ from a pool of unlabeled pairs U^t, and acquire the label $z_{a,b}^t$ from the oracle. We let $U^1 \subseteq X \times X$ be the initial pool of unlabeled pairs. Then $U^t = U^{t-1} \backslash (x_a^{t-1}, x_b^{t-1})$ for $1 \leq t \leq T$. Below we use $Z_t = [z_{a,b}^1, \cdots, z_{a,b}^t]$ to denote all the pair labels obtained up to the t-th iteration.

4.3.2.1 Selection Criteria.

We use two entropy-based criteria to select the best pair at each iteration. The first criterion, which we call *Uncertain*, is to select the pair whose label is the most uncertain. That is, at the t iteration, we choose the pair (x_a^t, x_b^t) that has the largest *marginal* entropy of $z_{a,b}^t$ (over the posterior distribution of W):

$$(x_a^t, x_b^t) \ = \ \arg \max_{(x_a, x_b) \in U^t} \ H(z_{a,b} | Z_{t-1}, X, \theta) \ . \tag{4.6}$$

Similar objective has been considered in prior work on distance metric learning [115] or document clustering [50], where the authors propose different approaches to compute/approximate the entropy objective.

The second criterion is a greedy objective adopted from active learning for classification [28, 43, 48], which we call *Info*. The idea is to select the query (x_a^t, x_b^t) that maximizes the marginal information gain about the model W:

$$\begin{aligned} (x_a^t, x_b^t) \ &= \ \arg \max_{(x_a, x_b) \in U^t} \ I(z_{a,b}, W | Z_{t-1}, X, \theta) \\ &= \ \arg \max_{(x_a, x_b) \in U^t} \ H(z_{a,b} | Z_{t-1}, X, \theta) - H(z_{a,b} | W, Z_{t-1}, X, \theta) \ . \end{aligned} \tag{4.7}$$

Note that here W is a random variable. The *Info* objective is equivalent to maximizing the entropy reduction about W, as can be proved by the chain rule of conditional entropy.

Interestingly, the first entropy term in the *Info* objective (4.7) is the same with the *Uncertain* objective (4.6). The additional term to *Info* is the conditional entropy of the pair label $z_{a,b}$ given W, i.e., the second term in (4.7). Comparing the two objectives,

we see that W is marginalized in the *Uncertain* objective and the selected query aims to reduce the maximum uncertainty of the *pair label*. In contrast, the goal of *Info* is to decrease the *model* uncertainty. There is subtle difference between these two types of uncertainties. The additional conditional entropy term in *Info* suggests that it prefers instance pairs whose labels are certain once W is known, yet whose overall uncertainty is high when marginalizing over W. In such sense, *Info* pays more attention to the uncertainty of the model W.

Each of the above selection objectives ranks the candidate pairs from the highest to the lowest. To select a pair to query, we go through the ranking and choose the one that does not create a cycle to the existing graph as described in Sec. 4.3.1. Since inference on trees are not only exact but also fast, enforcing such acyclic graph structure allows us to compute the selection objectives more effectively and accurately, and select more informative pairs to query.

4.3.2.2 Computing the Selection Objectives.

Now we describe how to compute the two objective values for a candidate instance pair. The two objectives require computing the marginal entropy $H(z_{a,b}|Z_t, X, \theta)$, and the conditional entropy $H(z_{a,b}|W, Z_t, X, \theta)$, for $1 \leq t \leq T$. By definition, the marginal entropy is

$$H(z_{a,b}|Z_t, X, \theta) = -\sum_{z_{a,b}} P(z_{a,b}|Z_t, X, \theta) \log P(z_{a,b}|Z_t, X, \theta) \ , \tag{4.8}$$

where the probability

$$P(z_{a,b}|Z_t, X, \theta) = \int P(\hat{W}|Z_t, X, \theta) P(z_{a,b}|Z_t, \hat{W}, X) d\hat{W} \ . \tag{4.9}$$

The conditional probability is computed as

$$P(z_{a,b}|Z_t, \hat{W}, X) = \frac{P(z_{a,b} \cup Z_t|\hat{W}, X)}{P(Z_t|\hat{W}, X)} \ , \tag{4.10}$$

where calculating both the numerator and the denominator are the same inference problem as (4.5) and can be solved similarly using message passing. In fact, message propa-

gations for the two calculations are shared except for that a new edge regarding $z_{a,b}$ is introduced to the graph for $P(z_{a,b} \cup Z_t|\hat{W}, X)$. So we can calculate the two values by performing message passing algorithm only once on the graph of $P(z_{a,b} \cup Z_t|\hat{W}, X)$, and record $P(Z_t|\hat{W}, X)$ in the intermediate step.

By definition, the conditional entropy is

$$H(z_{a,b}|W, Z_t, X, \theta) = \int P(\hat{W}|Z_t, X, \theta) H(z_{a,b}|Z_t, \hat{W}, X) d\hat{W} \ , \tag{4.11}$$

where $H(z_{a,b}|\hat{W}, Z_t, X)$ is also easy to compute once we know $P(z_{a,b}|Z_t, \hat{W}, X)$, which has been done in (4.10).

Now the only obstacle in calculating the two entropies is to take the expectations over the posterior distribution $P(W|Z_t, X, \theta)$ in (4.9) and (4.11). Here we use sampling to approximate such expectations. We first sample W's from $P(W|Z_t, X, \theta)$ and then approximate the expectations with the sample means. Directly sampling from the posterior at every iteration is doable but very inefficient. Below we describe a sequential MCMC sampling method ("particle filter") that effectively updates the samples of the posterior.

4.3.3 The Sequential MCMC Sampling of W

The main idea of the sequential MCMC method is to avoid sampling with random starts at every iteration by utilizing the particles obtained from the previous iteration.[1] Specifically, to obtain particles from distribution $P(W|Z_t, X, \theta)$, the sequential MCMC method first resamples from the particles previously sampled from $P(W|Z_{t-1}, X, \theta)$, and then performs just a few MCMC steps with these particles to prevent degeneration [42].

Here we maintain S particles in each iteration. We denote W_s^t, $1 \leq s \leq S$, as the s-th particle in the t-th iteration. For initialization, we sample particles $\{W_1^0, \cdots, W_S^0\}$ from the prior distribution $P(W|X, \theta)$ defined in (4.3) using slice sampling [75] [2], an MCMC method that can uniformly draw samples from an unnormalized density function. Since slice sampling does not require the target distribution to be normalized, the unknown constant in the prior (4.3) can be neglected here.

[1] Here we follow the convention of the particle filter field and call samples of W as "particles".

[2] Here we use the implementation `slicesample` provided in the MATLAB toolbox.

At iteration $t, 1 \leq t \leq T$, after a new pair label $z^t_{a,b}$ is observed, we perform the following two steps to update the particles and get samples from $P(W|Z_t, X, \theta)$.

(1) Resample. The first step is to resample from the particles $\{W_1^{t-1}, \cdots, W_S^{t-1}\}$ obtained from the previous iteration for $P(W|Z_{t-1}, X, \theta)$. We observe that

$$\begin{aligned} P(W|Z_t, X, \theta) &= P(W|z^t_{a,b}, Z_{t-1}, X, \theta) \\ &\propto P(z^t_{a,b}|Z_{t-1}, W, X)P(W|Z_{t-1}, X, \theta) \ . \end{aligned}$$

So each particle W_s^{t-1} is weighted by $P(z^t_{a,b}|Z_{t-1}, W_s^{t-1}, X)$, which can be calculated the same as (4.10).

(2) Move. In the second step, we start with each resampled particles, and perform several slice sampling steps for the posterior

$$P(W|Z_t, X, \theta) \propto P(Z_t|W, X)P(W|X, \theta) \ . \tag{4.12}$$

Again $P(Z_t|W, X)$ is calculated by message passing as (4.5), and the unknown normalizing constant in $P(W|X, \theta)$ can be ignored, since slice sampling does not require the normalization constant.

The *resample-move* method avoids degeneration in the sequence of slice sampling steps. After these two steps, we have updated the particles for $P(W|Z_t, X, \theta)$. Such particles are used to approximate the selection objectives as described in Sec. 4.3.2, allowing us to select the next informative pair to query.

Note that the distribution $P(W|Z_t, X, \theta)$ is invariant to *label switching*, that is, permuting column vectors of $W = [W_{\cdot,1}, \cdots, W_{\cdot,K}]$ will not change the probability $P(W|Z_t, X, \theta)$. This is because we can not provide any prior of W with label order, nor does the obtained constraints provide any information about the label order. One concern is whether the label switching problem would reduce sampling efficiency and affect the pair selection, since $P(W|Z_t, X, \theta)$ has multiple modes corresponding to different label permutations. Actually it does not cause an issue to the approximation of integrations in (4.9) and (4.11), since the term $P(z_{a,b}|Z_t, W, X, \theta)$ is also invariant to label permutations. However, the label switching problem does cause difficulty in get-

ting the Bayesian prediction of clusters labels from distribution $P(Y|Z_t, X, \theta)$, so we will employ the MAP solution W_{map} and predict cluster labels with $P(Y|Z_t, W_{map}, X, \theta)$. We describe this in the following section.

4.3.4 Find the MAP Solution

Given a set of constraints with pair labels Z, we first find the MAP estimation W_{map} by maximizing the posterior $P(W|Z, X, \theta)$, or equivalently maximizing the joint distribution $P(W, Z|X, \theta)$ (in the logarithmic form):

$$\max_{W} \quad L \;=\; \log P(W, Z|X, \theta) \;=\; \log P(Z|W, X) + \log P(W|X, \theta) \;. \tag{4.13}$$

The maximization can be solved by off-the-shelf gradient-based optimization approaches. Here we use the quasi-newton method provided in the MATLAB toolbox. The gradient of the objective L with respect to W is

$$\frac{\partial L}{\partial W} = \sum_{i \in \alpha(Z)} x_i (q_i - p_i)^\top - \lambda W - \frac{\tau}{N} \sum_{i=1}^{N} x_i \sum_{k=1}^{K} p_{ik} \log p_{ik} (\mathbf{1}_k - p_i)^\top \;,$$

where $p_i = [p_{i1}, \cdots, p_{iK}]^\top$ with $p_{ik} = P(y_i = k|W, x_i)$, $q_i = [q_{i1}, \cdots, q_{iK}]^\top$ with $q_{ik} = P(y_i = k|Z, W, x_i)$, and $\mathbf{1}_k$ is a K dimensional vector that contains 1 on the k-th dimension and 0 elsewhere. Here $\alpha(Z)$ again indexes all the instances involved in the constraints.

With the W_{map} solution to (4.13), we then find the MAP solution of the cluster labels Y from $P(Y|Z, W_{map}, X)$. This is done in two cases. For the instances that are *not* involved in the constraints, the MAP of Y is simply the most possible assignment of $P(Y|W_{map}, X)$. For the instances involved in the constraints, we need to find

$$\max_{Y_{\alpha(Z)}} \quad P(Y_{\alpha(Z)}|Z, W_{map}, X_{\alpha(Z)}) \propto P(Z|Y_{\alpha(Z)}) P(Y_{\alpha(Z)}|W_{map}, X_{\alpha(Z)}) \;.$$

The inference can be done by performing max-product algorithm on the same graph as defined for (4.5), only replacing the "summation" with the "max" operator at every message propagation.

In real applications, we only need to find the MAP solution of Y after the last

Algorithm 2 Bayesian Active Clustering with Pairwise Constraints

Input: data X, number of clusters K, access to the oracle, initial pool U^1, query budget T, prior parameter θ, number of samples S
Output: a clustering solution of the data

Initialize particles by sampling $\{W_1^0, \cdots, W_S^0\}$ from prior $P(W|X,\theta)$
for $t = 1$ **to** T **do**
1. **Select a pair to query:**
   Use particles $\{W_1^{t-1}, \cdots, W_S^{t-1}\}$ to compute the selection objective (4.6) or (4.7)
   Choose the best pair (x_a^t, x_b^t) from U^t and acquire $z_{a,b}^t$ from the oracle
2. **Update posterior:**
   Resample S particles with weight $P(z_{a,b}^t|Z_{t-1}, W_s^{t-1}, X)$ for W_s^{t-1}
   Perform a few MCMC steps on all particles with distribution $P(W|Z_t, X, \theta)$
3. **Update the pool:** $U^{t+1} \leftarrow U^t \backslash (x_a^t, x_b^t)$

end for
Find the MAP solution $W_{map} = \arg\max_W \log P(W|Z_T, X, \theta)$
Find the clustering solution $Y_{map} = \arg\max_Y \log P(Y|Z_T, W_{map}, X)$

iteration. In our experiments, we search for the solution at every iteration to show the performance of our method if we stop learning at any iteration. Our overall algorithm is summarized in Algorithm 2.

Note that an alternative of finding the clustering solution is to find the MAP of W and Y at the same time. However, we think our MAP estimation of W which marginalizes Y is more stable, and our calculation method is much simpler compared with the alternative.

4.4 Experiments

In this section, we empirically examine the effectiveness of the proposed method. In particular, we aim to answer the following questions:

- Is the proposed Bayesian clustering model effective at finding good clustering solutions with a small number of pairwise constraints?
- Is the proposed *active* clustering method more effective than state-of-the-art active clustering approaches?

Table 4.1: Summary of Active Clustering Datasets Information

Dataset	#Inst	#Dim	#Class	#Query
Fertility	100	9	2	60
Parkinsons	195	22	2	60
Crabs	200	5	2	60
Sonar	208	60	2	100
Balance	625	4	3	100
Transfusion	748	4	2	100
Letters-IJ	1502	16	2	100
Digits-389	3165	16	3	100

4.4.1 Dataset and Setup

We use 8 benchmark UCI datasets to evaluate our method. Table 4.1 provides a summary of the dataset information. For each dataset, we normalize all features to have zero mean and unit standard deviation.

We form the pool of unlabeled pairs using all instances in the dataset, and set the query budget to 60 for smaller datasets and to 100 for datasets with large feature dimension (e.g, *Sonar*) or larger dataset size. When a pair of instances is queried, the label is returned based on the ground-truth instance class/cluster labels. We evaluate the clustering results of all methods using pairwise F-Measure [13], which evaluates the harmonic mean of the precision and recall regarding prediction of instance pairwise relations. We repeat all experiments 30 times and average the results.

For the proposed Bayesian clustering model, we found that its performance is not sensitive to the values of the prior parameter τ or the ϵ used in the pair label distribution (4.4). Here we set $\tau = 1$ and $\epsilon = 0.05$, where the nonzero value of ϵ allows for "soft constraints". For the parameter λ, which controls the covariance of the Gaussian prior, we experimented with $\lambda \in \{1, 10, 100\}$ and found that $\lambda = 10$ is uniformly good with all datasets, which we fix as the default value. For each dataset, we maintain $S = 2dK$ samples of the posterior at every iteration.

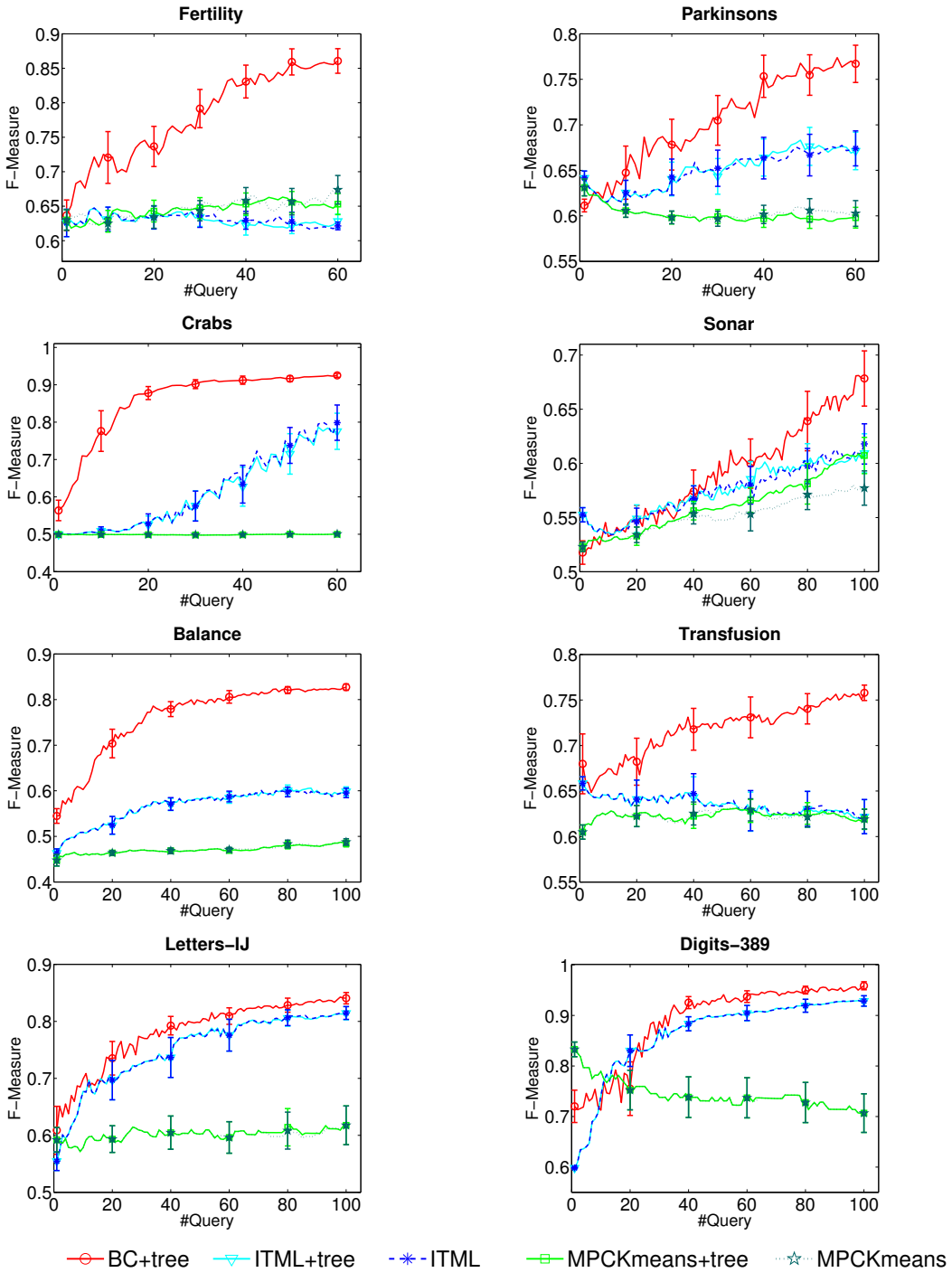


Figure 4.1: Pairwise F-Measure clustering results with increasing number of randomly selected queries. Results are averaged over 30 runs. Error bars are shown as mean and 95% confidence interval.

4.4.2 Effectiveness of the Proposed Clustering Model

To demonstrate the effectiveness of the proposed Bayesian clustering (BC) model, we compare with two well-known methods that learn from pairwise constraints: MPCK-means [13], and ITML [32][3]. In this set of experiment, we use randomly selected pairwise constraints to evaluate all methods. For our method, we incrementally select random pairs that do not introduce a cycle to the graph formed by existing pairs. To ensure a fair comparison, we evaluate ITML and MPCKmeans with randomly selected pairs with and without the acyclic graph restriction. Thus, all methods in competition are: *BC+tree*, *ITML*, *ITML+tree*, *MPCKmeans*, *MPCKmeans+tree*, where *BC+tree*, *ITML+tree*, and *MPCKmeans+tree* use randomly selected constraints that form a tree graph (or a forest), and *ITML* and *MPCKmeans* allow for cycles in the graph.

Figure 4.1 shows the performance of all methods with increasing number of constraints. We see that our method *BC+tree* outperforms the baselines on most datasets regardless of whether they use constraints with or without the acyclic graph restriction. This demonstrates the effectiveness of our Bayesian clustering model. We also notice that on most datasets we can hardly tell the difference between *ITML* and *ITML+tree*, or *MPCKmeans* and *MPCKmeans+tree*, suggesting that enforcing the acyclic structure in the constraints do not hurt the performance of ITML or MPCKmeans. Interestingly, such enforcement can in some cases produce better performance (e.g, on the *Sonar* dataset). We suspect this is because constraints forming cycles may have larger *incoherence* than those does not.[4] Davidson et al. [30] have shown that constraint sets with large incoherence can potentially degrade the clustering performance.

4.4.3 Effectiveness of the Overall Active Clustering Model

In this section, we compare our overall active clustering model with existing methods. Our baselines include two recent work on active learning with pairwise constraints: *Min-Max* [72], and *NPU* [111]. Both methods provide an active pair selection approach and require a clustering method to learn form the constraints. Here we supply them with

[3]ITML is a distance metric learning method, and we find the clustering solution by applying Kmeans clustering with the learned metric.

[4]The concept of *incoherence* is formally defined at [30]. Generally, a set of overlapping constraints tends to have higher incoherence than a set of disjoint constraints.

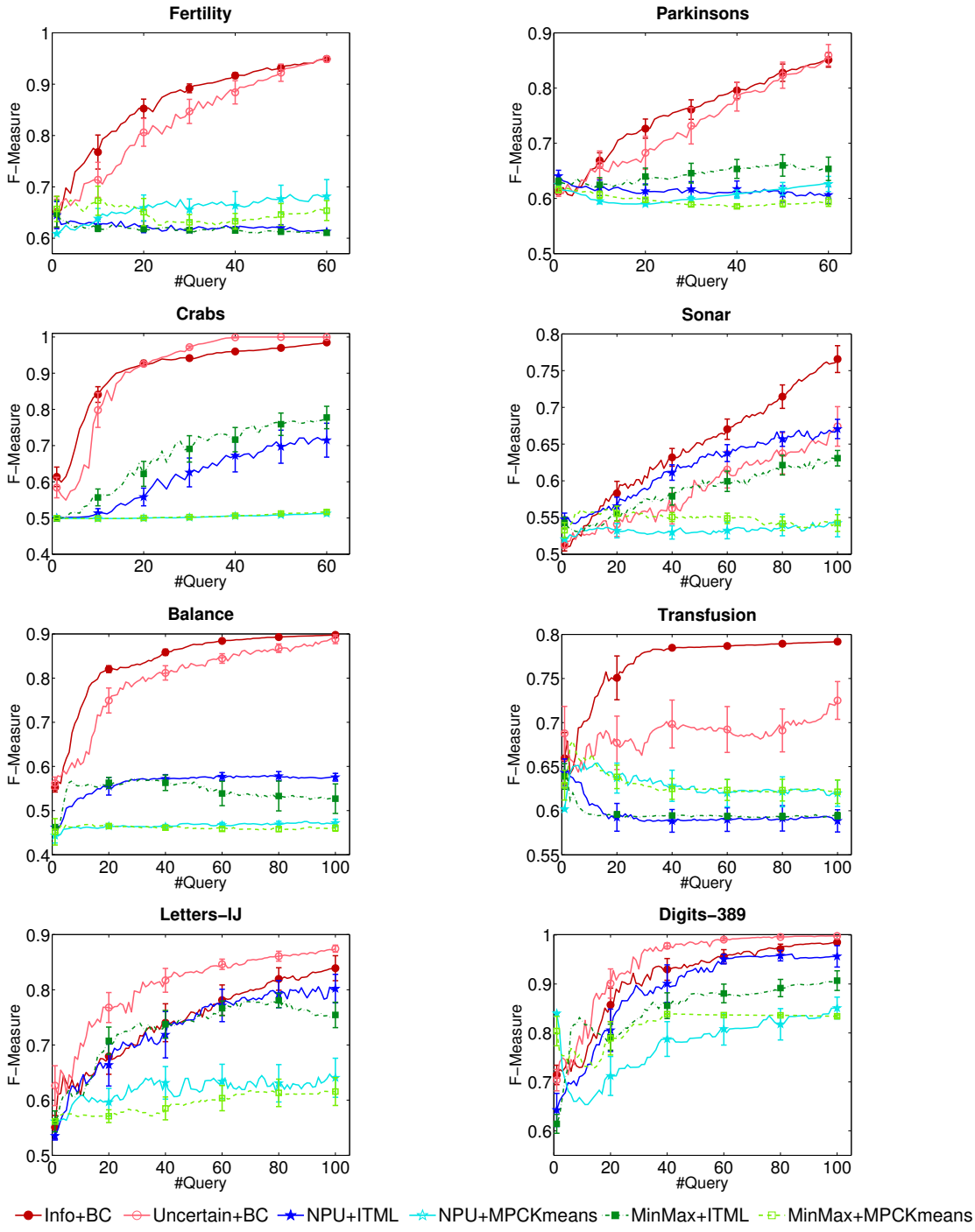


Figure 4.2: Pairwise F-Measure clustering results of different active clustering methods with increasing number of queries. Results are averaged over 30 runs. Error bars are shown as mean and 95% confidence interval.

MPCKmeans and ITML.[5] So all methods in competition are

- *Info+BC*: The proposed active clustering model with the *Info* criterion (4.7).
- *Uncertain+BC*: The proposed active clustering model with the *Uncertain* criterion (4.6).
- *NPU+ITML*: The *NPU* active selection strategy combined with *ITML*.
- *NPU+MPCKmeans*: The *NPU* method with *MPCKmeans*.
- *MinMax+ITML*: The *MinMax* active learning method combined with *ITML*.
- *MinMax+MPCKmeans*: The *MinMax* approach combined with *MPCKmeans*.

Figure 4.2 reports the performance of all active clustering methods with increasing number of queries. We see that both *Info+BC* and *Uncertain+BC* improve the clustering very quickly as more constraints are obtained, and they outperform all baselines on most datasets. Moreover, *Info+BC* seems to be more effective than *Uncertain+BC* in most cases. We hypothesize this is because *Info* reduces the uncertainty of the model, which might be more appropriate for improving the MAP solution of clustering than decreasing the maximum uncertainty of the pair labels as *Uncertain* does.

To avoid crowding Fig. 4.2, we did not present the passive learning results of our method *BC+tree* as a baseline in the same figure. The comparison between active learning and passive learning for our method can be done by comparing *Uncertain+BC* and *Info+BC* in Fig. 4.2 with *BC+tree* in Fig. 4.1. We see that both our active learning approaches produce better performance than passive learning on most datasets, demonstrating the effectiveness of our pair selection strategies.

We also notice that the performance of *NPU* or *MinMax* highly depends on the clustering method in use. With different clustering methods, their behaviors are very different. In practice, it can be difficult to decide which clustering algorithm should be used in combination with the active selection strategies to ensure good clustering performance. In contrast, our method unifies the clustering and active pair selection model, and the constraints are selected to explicitly reduce the clustering uncertainty and improve the clustering performance.

[5]Note that due to our Bayesian clustering model requires the set of constraints to form an acyclic graph, it can not be combined with *MinMax* or *NPU*, as they generally select constraints that form cycles due to their neighbourhood-based approach.

4.4.4 Analysis of the Acyclic Graph Restriction

Our method requires the graph formed by the constraints to be a tree (or a forest). Here we show that this restriction will not prevent us from selecting informative pairs. We examine the number of pairs that has been dropped at every iteration in order to find the best pair that does not create a cycle. Table 4.2 reports the results for the two selection criteria with varied number of queries. We see that for both criteria the number of dropped pairs is very small. For *Uncertain*, there is barely any pair that has been dropped on most datasets, and we see slightly more pairs dropped for the *Info* criteria. Overall, for only less than (often significantly less than) 10% of the number of queries, we encounter the need of dropping a pair. The only exception is the *Fertility* dataset, which is very small in size, making it difficult to avoid cycles with a large number of queries. But from the results in Sec. 4.4.3, we can see that the active clustering performance was still much better than the competing methods.

In addition, during our experiments, we found that for both criteria the difference between the maximum objective value and objective of the finally selected pair is often negligible. So in the case where some high-ranking pairs are dropped due to the acyclic graph structure restriction, the selected pair is still very informative. Overall, this enforcement does not present any significant negative impact on the final clustering results. It is interesting to note that, the results in Sec. 4.4.2 suggest that such graph structure restriction can in some cases improve the clustering performance.

4.5 Related Work

Prior work on active clustering for pairwise constraints has mostly focused on the neighbourhood-based method, where a neighbourhood skeleton is constructed to partially represent the underlying clusters, and constraints are queried to expand such neighbourhoods. Basu et al. [6] first proposed a two-phase method, Explore and Consolidate. The Explore phase incrementally builds K disjoint neighborhoods by querying instance pairwise relations, and the Consolidate phase iteratively queries random points outside the neighborhoods against the existing neighborhoods, until a must-link constraint is found. Mallapragada et al. [72] proposed an improved version, which modifies the Consolidate stage to query the most uncertain points using an MinMax objective. As

Table 4.2: Number of dropped pairs (shown as Info/Uncertain) required to find the best pair that does not a create cycle at different iterations. Results are averaged over 30 runs.

Dataset	Query Iteration					
	10	20	30	40	50	60
Fertility	0.4/0.0	0.6/0.1	0.9/0.1	2.7/1.9	4.2/14.3	10.8/32.0
Parkinsons	0.1/0.0	0.0/0.0	0.5/0.0	0.8/0.3	0.9/0.6	1.7/1.7
Crabs	0.6/0.0	0.2/0.0	0.0/0.0	0.1/0.3	0.2/0.6	0.4/1.5
Sonar	0.7/0.0	0.2/0.0	0.4/0.1	0.5/0.2	0.5/0.2	0.6/0.2
Balance	0.0/0.0	0.3/0.0	1.7/0.0	2.6/0.0	3.3/0.1	2.9/0.0
Transfusion	0.3/0.0	1.3/0.0	2.4/0.0	2.3/0.0	4.6/0.0	4.9/0.1
Letters-IJ	0.0/0.0	0.2/0.0	0.3/0.0	0.2/0.0	0.5/0.0	0.7/0.0
Digits-389	0.0/0.0	0.0/0.0	0.1/0.0	0.1/0.0	0.0/0.0	0.3/0.0

mentioned by Xiong et al. [111], these methods often select a batch of constraints before performing clustering, and they are not designed for iteratively improving clustering by querying new constraints, as considered in this work.

Wang and Davidson [105], Huang et al. [50] and Xiong et. al. [111] studied active clustering in an iterative manner. Wang and Davidson introduced an active spectral clustering method that iteratively select the pair that maximized the expected error reduction of current model. This method is however restricted to the two-cluster problems. Huang et al. proposed an active document clustering method that iteratively finds probabilistic clustering solution using a language model and they selected the most uncertain pair to query. But this method is limited to the task of document clustering. Xiong et. al. considered a similar iterative framework to Huang et al., and they queried the most uncertain data point against existing neighbourhoods, as apposed to the most uncertain pair in [50]. Xiong et al. only provide a query selection strategy and require a clustering method to learn from the constraints. In contrast, our method is a unified clustering and active pair selection model.

Finally, there are other methods that use various criteria to select pairwise constraints. Xu et al. [112] proposed to select constraints by examining the spectral eigenvectors of the similarity matrix in the two-cluster scenario. Vu et al. [101] proposed to select constraints involving points on the sparse regions of a k-nearest neighbours graph. The work [1, 46] used ensemble approaches to select constraints. The scenarios

considered in these methods are less similar to what has been studied in this chapter.

4.6 Conclusion

In this chapter, we studied the problem of active clustering, where the goal is to iteratively improve clustering by querying informative pairwise constraints. We introduced a Bayesian clustering method that adopted a logistic clustering model and a data-dependent prior which controls model complexity and encourages large separations among clusters. Instead of directly computing the posterior of the clustering model at every iteration, our approach maintains a set of samples from the posterior. We presented a sequential MCMC method to efficiently update the posterior samples after obtaining a new pairwise constraint. We introduced two information-theoretic criteria to select the most informative pairs to query at every iteration. Experimental results demonstrated the effectiveness of the proposed Bayesian active clustering method over existing approaches.

Manuscript 4: Learning with Latent Label Hierarchy from Incomplete Multi-Label Data

Yuanli Pei, Xiaoli Z. Fern, and Raviv Raich

Chapter 5: Learning with Latent Label Hierarchy from Incomplete Multi-Label Data

Abstract

Exploiting label structure for multi-label classification can significantly improve classification performance and also benefit the labeling process. One useful type of structure is the label hierarchy. Existing work either can not make use of such structure or assume the hierarchy is given as a prior. In practice, such hierarchy is not always available beforehand and it is desirable to learn it from data. Moreover, the label assignments in the training data may be incomplete due to inconsistencies in the labeling process, which raises another challenge for learning. In this chapter, we study multi-label learning with latent label hierarchy and incomplete label assignments. Our goal is to simultaneously learn the hierarchy as well as a multi-label classifier given the input features and incomplete label assignments. We propose a probabilistic model that captures the hierarchical structure and the incompleteness of the labels. Based on this model, we employ a maximum likelihood estimation approach and use Expectation-Maximization (EM) to optimize the objective. In the E-step, we propose an efficient dynamic programming approach to compute the posterior. In the M-step, we estimate the classifier parameter and formulate the problem of learning the hierarchy as a maximum spanning tree problem.

5.1 Introduction

In multi-label learning, each data instance is associated with a set of features and a collection of labels. Multi-label data exists in various applications, including image classification, music categorization, and bioinformatics, to name a few. For example, an image can belong to multiple semantic scenes [130], a song may belong to more than one genres [63], and a gene may be associated with a number of functions [5]. In many applications, the label occurrences are not independent of each other. Thus, one of the goals in multi-label learning is to model the underlying structure of the labels.

Among various structures, the label hierarchy is a practical and useful type. For example, in one of our motivating applications, the data involves reviews of events and

venues and tagging of them from multiple resources. For a dinning venue, the possible tags may include "Asian Restaurant", "Chinese Restaurant", "Sichuan Restaurant", and "Japanese Restaurant", etc. Leveraging the hierarchical structure of the labels, one can constrain the classifier to follow the rule that a "Sichuan Restaurant" is also a "Chinese Restaurant", and it should not be a "Japanese Restaurant", thus improving the classification. Several studies have shown that when provided as a prior, such a label hierarchy can help to learn a classifier with significantly improved classification performance [10, 84, 85, 93, 100].

Moreover, such label hierarchies are not only useful for classification, but also beneficial to the labeling process. Consider the previous example, suppose a hierarchy is available regarding the restaurant labels. When an annotator labels a restaurant as "Sichuan Restaurant", the labels "Chinese Restaurant" and "Asian Restaurant" could be recommended as potential classes or automatically completed according to the hierarchy. This will significantly increase the labeling efficiency as well as consistency.

While the label hierarchy is usually assumed as a prior by existing work, it is not always available beforehand, and it is more desirable to learn it from data. One challenge in learning the hierarchy is that the label assignments in the training data may be incomplete due to inconsistent labeling processes from one or multiple sources. Consider our previous example again, one annotator may tag a restaurant as a "Chinese Restaurant" as well as an "Asian Restaurant", while another annotator might only tag it as an "Asian Restaurant", i.e., the label "Chinese Restaurant" is missing. Such inconsistency in the labeling process naturally occurs when multiple annotators are involved, resulting in incomplete assignments for the training data and creating challenges for learning.

In this work, we consider the problem of learning with latent label hierarchy from incomplete multi-label data. Our goal is to simultaneously discover the latent label hierarchy and learn a multi-label classifier that is consistent with the hierarchy. Intuitively, knowing the label hierarchy can help improve classification accuracy. Inversely, a good classifier can also help better infer the hierarchy. There are two challenges in this problem. First, the learning space for the underlying label hierarchy is exponentially large in the number of classes, which makes it difficult to learn the hierarchy. Second, as stated previously, the label incompleteness issue exists which makes the problem even more difficult.

Here we propose a graphical model that captures the labeling process. By specifying

appropriate conditional probabilities, we capture the label hierarchy structure and the property of incomplete label assignments. Based on this model, we use the maximum likelihood approach to estimate the parameters and learn the structure. Due to the latent variables (the underlying complete label assignments), we derive an Expectation Maximization (EM) solution for optimization. In particular, we propose a dynamic programming approach to infer the posterior distributions in the E-step. In the M-step, we estimate the parameters for the classifier and formulate the problem of learning the label hierarchy as a maximum spanning tree problem, which can be solved efficiently.

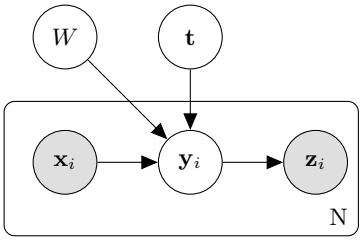
We empirically evaluate our method on real-world datasets from various applications. The results demonstrate that our method can generally learn a better label hierarchy than the baseline methods, and a competitive multi-label classifier. In addition, we perform a case study on a real-world dataset without ground-truth structure and discover a meaningful hierarchy for the labels.

5.2 Problem Statement

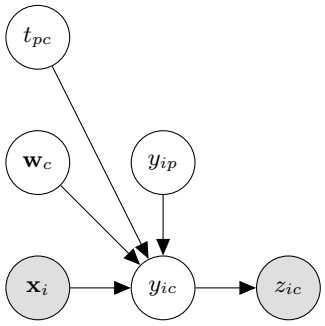
In this problem, we are given a dataset $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top, \mathbf{x}_i \in \mathcal{R}^d$, and the observed incomplete label set $Z = [\mathbf{z}_1, \ldots, \mathbf{z}_N]^\top$, where $\mathbf{z}_i = [z_{i1}, \ldots, z_{iK}]$ is the K dimensional binary label vector of $\mathbf{x}_i$, and d is the feature dimension and K is the total number of classes. The observed labels Z are noisy observations of the underlying labels $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^\top$ with $\mathbf{y}_i = [y_{i1}, \ldots, y_{iK}]$, in a way that if the underlying label is positive, it could be missing and observed as negative. In this problem, the K classes form an unknown label hierarchy, and the underlying Y is consistent with this hierarchy. That is, if a child class is positive, then all the ancestor classes are also positive. However, the incomplete assignments Z may not be consistent with the hierarchy since they are noisy observation of the underlying labels.

To define the hierarchy, we let $\mathbf{t} \in \mathcal{T}$ be an indicator vector of $2\binom{K}{2}$ dimensional, where $t_{pc} \in \{0, 1\}$ indicates whether the class p is a parent of class c, for $(p, c) \in [K] \times [K]$ [1], and the notation $[K] = \{1, \ldots, K\}$ is the set of integers ranging from 1 to K. Note that the multiplier 2 before $\binom{K}{2}$ for the dimension is due to the two possible edge directions for each pair of nodes/classes. The feasible space $\mathcal{T}$ contains all the possible trees connecting the K class nodes.

[1] For notation simplicity, we define $t_{c,c} = 0$ and do not learn them.



(a) The Overall Graphical Model



(b) Dependencies regarding label z_{ic} with $t_{pc} = 1$. Other dependencies not related to z_{ic} are omitted.

Figure 5.1: A graphical model for learning with latent label hierarchy from incomplete multi-label data.

Given the setup, our goal is to simultaneously learn the label hierarchy defined by $\mathbf{t}$, as well as a multi-label classifier parameterized by W that is consistent with $\mathbf{t}$.

5.3 The Probabilistic Graphical Model

We propose the graphical model illustrated in Figure 5.1 to capture the dependencies between the input features X, the underlying labels Y that are consistent with the hierarchy $\mathbf{t}$, and the observed incomplete labels Z.

Figure 5.1(a) shows the overall structure of the graphical model. The hierarchy vector $\mathbf{t}$ determines the edge connections among the elements in $\mathbf{y}_i$. The underlying label $\mathbf{y}_i$ depends on the features $\mathbf{x}_i$, the parameter W, and the ancestor labels in $\mathbf{y}_i$ indicated by $\mathbf{t}$. The observed labels $\mathbf{z}_i$ only depend on the underlying label vector $\mathbf{y}_i$.

Here we make two assumptions. First, the underlying label y_{ic} is conditionally independent from the other y_{ik}'s for $k \neq p$, given its parent node y_{ip} the features $\mathbf{x}_i$, and the parameter $\mathbf{w}_c$ for class c. Secondly, the observation z_{ic} depends only on the underlying label y_{ic}.

To illustrate how the hierarchy structure $\mathbf{t}$ determines the dependencies among $\mathbf{y}$'s, Figure 5.1(b) shows the model regarding z_{ic} with $t_{pc} = 1$, indicating that class p is a parent of class c. Other variables that are not related to z_{ic} are omitted.

Table 5.1: Specification of conditional probabilities for learning with latent label hierarchy from incomplete multi-label data.

(a) Conditional probability $P(y_{ic}|\mathbf{x}_i, \mathbf{w}_c, y_{ip})$.

	$y_{ip} = 0$	$y_{ip} = 1$
$P(y_{ic} = 0\|\mathbf{x}_i, \mathbf{w}_c, y_{ip})$	$1 - \epsilon$	$1 - \sigma(\mathbf{w}_c^\top \mathbf{x}_i)$
$P(y_{ic} = 1\|\mathbf{x}_i, \mathbf{w}_c, y_{ip})$	ϵ	$\sigma(\mathbf{w}_c^\top \mathbf{x}_i)$

(b) Conditional probability $P(z_{ic}|y_{ic}, \alpha)$

	$y_{ic} = 0$	$y_{ic} = 1$
$P(z_{ic} = 0\|y_{ic}, \alpha)$	1	α
$P(z_{ic} = 1\|y_{ic}, \alpha)$	0	$1 - \alpha$

5.3.1 Model Specification

To capture the property that the underlying labels are consistent with the latent hierarchy, we design the conditional distribution $P(y_{ic}|\mathbf{x}_i, \mathbf{w}_c, y_{ip})$ as shown in Table 5.1(a). Specifically, if an instance does not belong to the underlying parent class, then there is a very small probability (controlled by ϵ) that it belongs to the child class. Such relaxation is introduced to accommodate the labeling inconsistency in the data as well as allowing the learning process to be "soft" when necessary. Otherwise if the the parent class is positive, we use a logistic model $\sigma(\cdot)$ [14] to determine whether the instance belongs to the child class based on the features $\mathbf{x}_i$ and the corresponding parameter $\mathbf{w}_c$. With this definition, the conditional probability of the complete label set $\mathbf{y}_i$ is factorized as

$$P(\mathbf{y}_i|\mathbf{x}_i, W, \mathbf{t}) = P(y_{i,root}) \prod_{(p,c)} P(y_{ic}|\mathbf{x}_i, \mathbf{w}_c, y_{ip})^{t_{pc}} \tag{5.1}$$

where we set $P(y_{i,root} = 1) = 1$ for the root variable [2].

To deal with the problem that the observed labels $\mathbf{z}$'s are incomplete, we employ

[2]Note that when the underlying structure is a forest, we can always add a dummy root variable to form a tree structure. Here we simply add a dummy root for all cases.

the conditional probability $P(z_{ic}|y_{ic})$ as shown in 5.1(b). Here if the underlying label is negative, then the observed labels are always negative. Otherwise if the underlying label is positive, there is a chance (reflected by α) that the observed label is negative, namely, the label could be missing. Here we estimate the missing factor α from the data.

5.3.2 Maximum-Likelihood Objective

Our objective is to maximize the likelihood of the observed labels Z given the parameters W and α, and the unknown structure $\mathbf{t}$. That is,

$$\begin{aligned} \max_{W,\mathbf{t},\alpha} \quad & \log P(Z|X,W,\mathbf{t},\alpha) - \tfrac{\lambda}{2d}\|W\|_2^2 \\ = \quad & \sum_i \log \sum_{\mathbf{y}_i} P(\mathbf{y}_i,\mathbf{z}_i|\mathbf{x}_i,W,\mathbf{t},\alpha) - \tfrac{\lambda}{2d}\|W\|_2^2 \ . \end{aligned} \tag{5.2}$$

where we add an additional regularization term for the parameter W, and d is the feature dimension. The objective involves marginalizing over the underlying labels Y, which is difficult to compute. Below we use the Expectation-Maximization (EM) approach to optimize the objective.

5.4 Optimization with Expectation-Maximization

The Expectation-Maximization (EM) approach [14] alternates between finding a lower bound of the objective in the E-steps, and maximizing the lower bound in the M-steps.

To find the lower bound, we apply Jensen's inequality and obtain

$$\begin{aligned} \max_{W,\mathbf{t},\alpha} \quad & \sum_i \log \sum_{\mathbf{y}_i} P(\mathbf{y}_i,\mathbf{z}_i|\mathbf{x}_i,W,\mathbf{t},\alpha) - \tfrac{\lambda}{2d}\|W\|_2^2 \\ \geq \quad & \sum_i E_{Q(\mathbf{y}_i)}[\log P(\mathbf{y}_i|\mathbf{x}_i,W,\mathbf{t}) + \log P(\mathbf{z}_i|\mathbf{y}_i,\alpha)] + \sum_i H[Q(\mathbf{y}_i)] - \tfrac{\lambda}{2d}\|W\|_2^2 \ . \end{aligned} \tag{5.3}$$

Here $Q(\mathbf{y}_i)$ is a distribution over $\mathbf{y}_i$, and $H[Q(\mathbf{y}_i)] = -E_{Q(\mathbf{y}_i)}[\log Q(\mathbf{y}_i)]$ is the entropy of $Q(\mathbf{y}_i)$. The lower bound is tight when $Q(\mathbf{y}_i)$ equals the posterior $P(\mathbf{y}_i|\mathbf{z}_i,\mathbf{x}_i,W,\mathbf{t},\alpha)$ for $i \in [N]$.

In E-steps, we find the posterior distribution of the hidden variables Y for computing the lower bound. In M-steps, we maximize over the W, α and $\mathbf{t}$ with respect to the lower bound. Below, we describe our E-steps and M-steps.

5.4.1 E-Step

In the E-step, we need to compute the posterior $Q(\mathbf{y}_i) = P(\mathbf{y}_i|\mathbf{z}_i, \mathbf{x}_i, W, \mathbf{t}, \alpha)$ for $i \in [N]$. However, the dimension of $Q(\mathbf{y}_i)$ is 2^K, which is too expensive to compute. Interestingly, we observe that the posterior is marginalized in the M-step to obtain the pairwise posterior. Thus, here in the E-step we only compute the pairwise posterior $Q(y_{ip}, y_{ic})$ for each pair $(p, c) \in [K] \times [K]$. In this way, the dimension of the posterior distribution $Q(\mathbf{y}_i)$ is reduced to $2 \times 2 \times \binom{K}{2}$.

To compute each $Q(y_{ip}, y_{ic})$, we need to marginalize over all the other labels over $P(\mathbf{y}_i|\mathbf{z}_i, \mathbf{x}_i, W, \mathbf{t}, \alpha)$. That is,

$$\begin{aligned} Q(y_{ip}, y_{ic}) &= P(y_{ip}, y_{ic}|\mathbf{z}_i, \mathbf{x}_i, W, \mathbf{t}, \alpha) \\ &= \frac{\sum_{\mathbf{y}_i \setminus p,c} P(\mathbf{y}_i, \mathbf{z}_i|\mathbf{x}_i, W, \mathbf{t}, \alpha)}{\sum_{\mathbf{y}_i} P(\mathbf{y}_i, \mathbf{z}_i|\mathbf{x}_i, W, \mathbf{t}, \alpha)} . \end{aligned}$$

Brute force marginalization of $P(\mathbf{y}_i, \mathbf{z}_i|\mathbf{x}_i, W, \mathbf{t})$ for every pair is too expensive. Below we propose an efficient method to compute the marginalization.

We first divide the pairs into two categories: the pairs that are *connected* by an edge on the tree, and the pairs that are *not connected* by an edge on the tree. We describe efficient ways of performing marginalization for each category.

For the pairs that are connected by an edge on the tree, we can simply run the message passing algorithm [14] on the factor graph formed by the class nodes on the tree. Specifically, the node potentials and edge potentials can be identified by noticing that

$$\begin{aligned} P(\mathbf{y}_i, \mathbf{z}_i|\mathbf{x}_i, W, \mathbf{t}, \alpha) = &\ P(\mathbf{z}_i|\mathbf{y}_i, \alpha) P(\mathbf{y}_i|\mathbf{x}_i, W, \mathbf{t}) \\ = &\ \prod_{k=1}^{K} \underbrace{P(z_{ik}|y_{ik}, \alpha)}_{\text{node potential}} \prod_{(p,c)} \underbrace{P(y_{ic}|\mathbf{x}_i, \mathbf{w}_c, y_{ip})^{t_{pc}}}_{\text{edge potential}} . \end{aligned} \tag{5.4}$$

As message passing is exact on trees, the pairwise posteriors are computed exactly.

For the pairs that are not connected by an edge, we propose a dynamic programming approach to efficiently compute the posteriors. The key idea is to use the conditional independence given the parent node on the tree. Specifically, we first find an ordering of the nodes such that the parent nodes always appear earlier than their children. For example, this can easily be computed via a Breadth-First Search (BFS) or a Depth-First Search (DFS) ordering. Then, for each node k in the ordering, we compute the posterior distributions for all the related pairs (k, c) that have not been computed, following the same ordering. For each pair (k, c), suppose p is the parent of c. We can efficiently compute the pairwise posterior (k, c) for instance i as below

$$\begin{aligned} Q(y_{ik}, y_{ic}) = & \sum_{y_{ip}} Q(y_{ic}|y_{ik}, y_{ip}) Q(y_{ik}, y_{ip}) \\ = & \sum_{y_{ip}} Q(y_{ic}|y_{ip}) Q(y_{ik}, y_{ip}) \\ = & \sum_{y_{ip}} \frac{Q(y_{ip}, y_{ic})}{Q(y_{ip})} Q(y_{ik}, y_{ip}) \ , \end{aligned} \tag{5.5}$$

where the second step follows because of the conditional independence between y_{ic} and y_{ik} given the parent y_{ip}. Here since $p \prec c$ in the ordering, $Q(y_{ik}, y_{ip})$ is computed before $Q(y_{ik}, y_{ic})$. The distributions $Q(y_{ip}, y_{ic})$ and $Q(y_{ip})$ are also computed if the first categories of the pairs are processed first. Thus, we only need to marginalize over one node y_{ip} instead of $K-2$ nodes, which significantly reduces the computation complexity.

In short, we first run message passing on the tree to obtain the marginal distributions $Q(y_{ip})$ and pairwise distributions $Q(y_{ip}, y_{ic})$. This process calculates all the posterior for the first category of pairs. We then compute the posteriors of the second category of pairs using (5.5) in an ordering described above. We summarize the E-step in Algorithm 3.

Algorithm 3 E-step: Compute $Q(\mathbf{y}_i) = P(\mathbf{y}_i|\mathbf{z}_i, \mathbf{x}_i, W, \mathbf{t}, \alpha)$

Input: Data $\mathbf{x}_i$, observed label vector $\mathbf{z}_i$, parameter W, α, and tree structure $\mathbf{t}$.
Output: Posterior $Q(y_{ik})$ for $k = [K]$, and $Q(y_{ip}, y_{ic})$ for all possible pairs (p, c).

Form the factor graph from $\mathbf{t}$ using the node and edge potentials defined by (5.4).
Run message passing to find the marginal distributions $Q(y_{ic})$ for each node c and $Q(y_{ip}, y_{ic})$ for each edge (p, c).
Find a BFS or DFS ordering. Suppose the nodes are ordered from 1 to K.
for $k = 1, \ldots, K$ **do**
  for $c = k + 1, \ldots, K$ **do**
    if $t_{kc} == 1$ **then**
      Pass. // An edge exists and $Q(y_{ik}, y_{ic})$ is already computed.
    else
      Compute $Q(y_{ik}, y_{ic})$ using (5.5).
    end if
  end for
end for

Algorithm 4 EM Algorithm for Learning with Latent Label Hierarchy from Incomplete Multi-Label Data

Input: Dataset $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$, observed label set $Z = [\mathbf{z}_1, \ldots, \mathbf{z}_N]$.
Output: Classifier Parameter W, label missing factor α, and label structure $\mathbf{t}$.

Initialize W, $\mathbf{t}$, and α .
repeat
  E-step:
    Compute $Q(\mathbf{y}_i) = P(\mathbf{y}_i|\mathbf{z}_i, \mathbf{x}_i, W, \mathbf{t}, \alpha)$ for all $i \in [N]$ using Algorithm 3.
  M-step:
    Update α using (5.7)
    repeat
      Optimize W using gradient base approach with the gradient defined by (5.8)
      Optimize $\mathbf{t}$ by finding a maximum spanning tree on a weighted directed complete graph with the edge weights defined by (5.9)
    until stopping criteria
until convergence

5.4.2 M-Step

In the M-step, given the posterior distribution $Q(\mathbf{y}_i)$, we need to optimize the lower bound (5.3) with respect to W, $\mathbf{t}$, and α. The related objective is

$$\begin{aligned}\max_{W,\mathbf{t},\alpha} \quad & \sum_i E_{Q(\mathbf{y}_i)}[\log P(\mathbf{y}_i, \mathbf{z}_i|\mathbf{x}_i, W, \mathbf{t}, \alpha)] - \tfrac{\lambda}{2d}\|W\|_2^2 \\ = \quad & \sum_i \sum_{\mathbf{y}_i} Q(\mathbf{y}_i) \log P(\mathbf{y}_i|\mathbf{x}_i, W, \mathbf{t}) + \sum_i \sum_{\mathbf{y}_i} Q(\mathbf{y}_i) \log P(\mathbf{z}_i|\mathbf{y}_i, \alpha) - \tfrac{\lambda}{2d}\|W\|_2^2 \ .\end{aligned} \tag{5.6}$$

We show that there is a closed-form solution for α, and we can iteratively optimize over W and $\mathbf{t}$ with efficient method for learning the structure $\mathbf{t}$.

Optimize α: To optimize over α, only the second term in (5.6) is related. Substituting the conditional probabilities in Table 5.1(b), we have the following solution

$$\alpha^* = 1 - \sum_i \sum_k \frac{Q(y_{ik} = 1) z_{ik}}{\sum_i \sum_k Q(y_{ik} = 1)} \ . \tag{5.7}$$

Intuitively, the estimation can be viewed as the inconsistency between the posterior and the observed values. The more consistent the z_{ik} is with the marginal posterior $Q(y_{ik})$, the smaller the α value is.

Optimize W and $\mathbf{t}$: To optimize W and $\mathbf{t}$, only the second term and third in (5.6) is related. Here we alternatively optimize them.

- Given $Q(\mathbf{y}_i)$, α and fixed structure $\mathbf{t}$, we can optimize W using gradient based approach such as L-BFGS. The gradient for each $\mathbf{w}_c$ is given by

$$\nabla_{\mathbf{w}_c} = \sum_i \sum_{(p,c)} t_{pc} Q(y_{ip} = 1, y_{ic} = 1) \Big\{ 1 - [1 + \frac{Q(y_{ip} = 1, y_{ic} = 0)}{Q(y_{ip} = 1, y_{ic} = 1)}] \sigma(\mathbf{w}_c^\top \mathbf{x}_i) \Big\} \mathbf{x}_i - \frac{\lambda}{d} \mathbf{w}_c \ . \tag{5.8}$$

- Learning the tree structure $\mathbf{t}$ is generally a difficult problem. Interestingly, here we show that we can formulate this problem as a maximum spanning tree problem, which can be solved efficiently. In particular, the related objective regarding

Table 5.2: Summary of Multi-Label Datasets Information.

dataset	#Data	#Feats	#Labels
Pascal 2007	9963	4097	24
20 Newsgroup	2000	100	27
Church-FUN	302	20	16

optimizing $\mathbf{t}$ is

$$\max_{\mathbf{t}\in\mathcal{T}} \quad \sum_{(p,c)} t_{pc} \underbrace{\Big[\sum_{i} \sum_{(y_{ip},y_{ic})} Q(y_{ip}, y_{ic}) \log P(y_{ic}|\mathbf{x}_i, \mathbf{w}_c, y_{ip})\Big]}_{\text{weight on edge (p,c)}} . \tag{5.9}$$

We can see that this problem essentially finds a maximum spanning tree on a complete directed weighted graph, with the edge weights defined by (5.9). Thus, we can find the structure efficiently using Chu-Liu/Edmond's algorithm [26, 39].

We summarize the overall EM optimization in Algorithm 4.

5.5 Experiments

In this section, we extensively evaluate the effectiveness of the proposed method. In particular, we would like to answer the following questions:

- Can our model learn a reasonable label hierarchy from incomplete multi-label data?
- Does the learned multi-label classifier outperform other methods for multi-label classification with incomplete label assignments?

We investigate these two questions in the following two subsections, and then apply our method to a case study.

5.5.1 Learning Latent Label Hierarchy

In this set of experiment, we evaluate the effectiveness of our model in learning the label hierarchy from incomplete multi-label data.

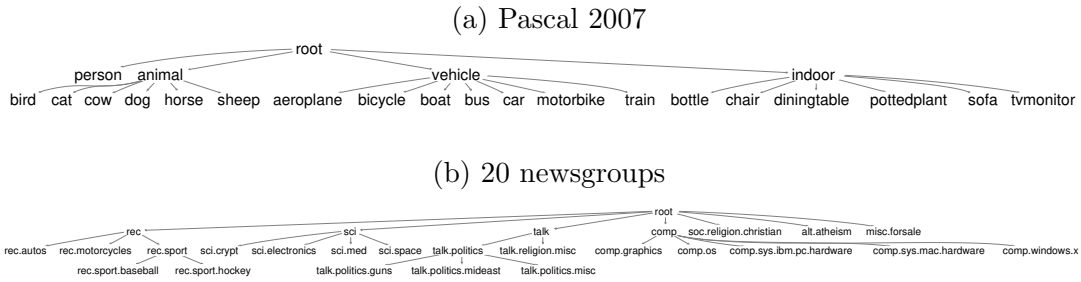


Figure 5.2: Ground-truth Label Hierarchy of the Pascal 2007 and 20 Newsgroup datasets.

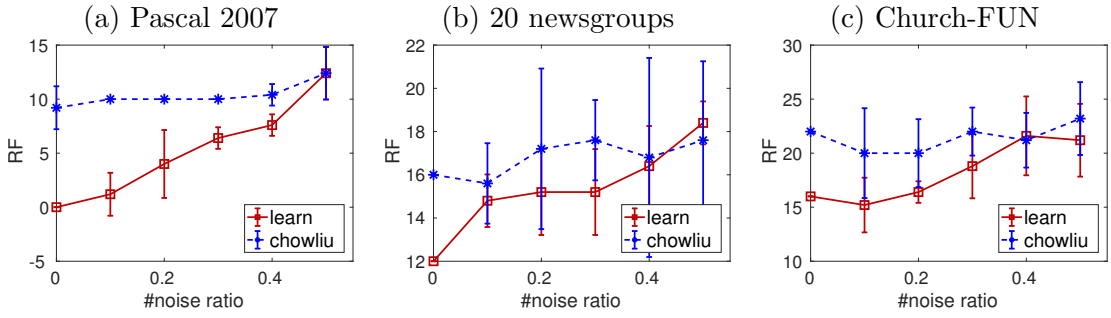


Figure 5.3: RF measure of the learned tree structure with different label noise ratios. Error bars are 95% confidence interval.

Datasets. We evaluate our method on three benchmark multi-label datasets with ground-truth label hierarchy: the *Pascal 2007* dataset[3], the *20 Newsgroup* dataset[4], and the *Church-FUN* dataset [5].

The *Pascal 2007* dataset is an image classification dataset, which originally has 20 labels belonging to four semantic groups: "person", "animal", "vehicle", and "indoor". We explicitly add such semantic labels and assign positive labels to an instance if any of the children labels is positive. To represent the images, we extract CNN features using the tool OverFeat[6].

The *20 Newsgroup* dataset is a document dataset that contains subject topics from different domains. We form the label hierarchy by creating labels for higher abstraction levels of the topics. For example, from "rec.sport.baseball", we derive the higher level labels "rec" and "rec.sport". We assign the such labels positive if any of the subtopics is positive for an instance. We reduce the dimension of the bag-of-words features into 100 using principle component analysis.

The *Church-FUN* dataset is a subset of the gene dataset introduced in [100]. We extracted a subtree from the original dataset that contains relative large number of training instances.

The summary of all the dataset information is listed in Table 5.2. We show the ground-truth label hierarchies of the *Pascal 2007* and *20 Newsgroup* datasets in Figure 5.2.

Baseline Methods. We are not aware of any work that can jointly learn a tree structure as well as the classifier from incomplete multi-label data. The most reasonable method that is applicable to this problem is the ChowLiu method [26]. It first forms a complete graph of the labels, with the edge weights defined by the mutual information between each pair of nodes/classes, and then finds a maximum spanning tree on the complete graph. In this way, ChowLiu method can not make use of the input features. Thus, we only supply ChowLiu method with the labels of the data.

Experiment Setup. To evaluate how well our method can learn the hierarchy from incomplete multi-label data, we create synthetic label noise to the training data. In

[3] `http://host.robots.ox.ac.uk/pascal/VOC/voc2007/`
[4] `http://qwone.com/~jason/20Newsgroups/`
[5] `https://dtai.cs.kuleuven.be/clus/hmcdatasets/`
[6] `http://cilvr.nyu.edu/doku.php?id=code:start#overfeat`

particular, if a label for an instance is positive, then there is a chance that the label is missing and observed as negative. We vary the noise factor from 0.1 to 0.5 with a 0.1 increment. We input both the instance features and the incomplete labels to our method, and only supply Chowliu method with the label set. We perform 5 randomized runs and report the averaged results.

Performance Measure. We measure the learned the tree structure against the ground-truth structured using Robinson Foulds metric (RF) [82]. This popular metric computes the number of graph transformations (edge contraction or expansion) needed to change the estimated graph into the correct structure. In particular, let $\Sigma(T)$ be the set of edges in tree T. The RF distance between the ground truth tree T^* and the learned tree $\hat{T}$ is defined as

$$RF(T^*, \hat{T}) = |\Sigma(T^*) - \Sigma(\hat{T})| + |\Sigma(\hat{T}) - \Sigma(T^*)| \ .$$

That is, the RF distance between $\hat{T}$ and T^* is the sum of the number edges in T^* but not in $\hat{T}$, and the number of edges in $\hat{T}$ but not in T^*. The lower the metric is, the better that the underlying tree T^* is recovered by the learned structure $\hat{T}$.

Implementation Details. For the regularization parameter, we set $\lambda = \{0.01, 1, 100\}/d$ for each dataset, and pick the parameter with the best performance. For the parameter ϵ that controls the relaxation of the hierarchy consistency in Table 5.1(a), we start with the value $\epsilon = 0.05$, and then gradually decrease it by a factor of 0.98 at each iteration. The intuition is that the hierarchy learned during the early iterations may not be very accurate, and we should allow for more relaxation. As the optimization proceeds, the relaxation should decrease so that the classifier is more consistent with the hierarchy.

Results. Figure 5.3 shows the RF results of the learned tree structure with different label noise ratios. We can see that, in general, our method (denoted as *learn*) can learn a better tree structure compared to ChowLiu. In particular, on the Pascal 2007 dataset, our method can almost perfectly recover the structure when there is little noise in the data, while the Chowliu method was not able to achieve such result by only using the labels. This suggests that the instance features provide additional information that helps to recover a better tree structure from the incomplete labels.

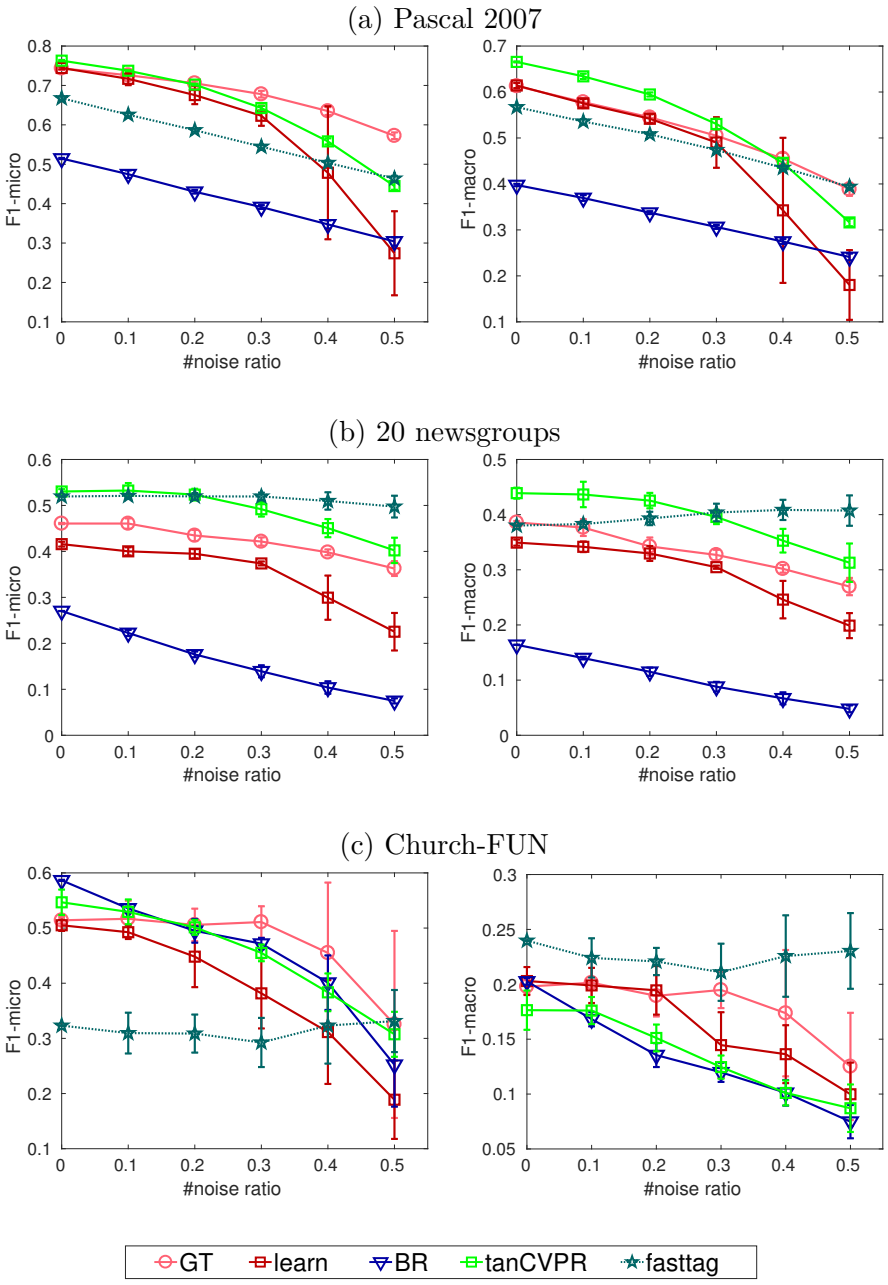


Figure 5.4: F1-Micro and F1-Macro multi-label classification performance with different label noise ratios.

5.5.2 Classification with Incomplete Multi-Label Data

In this set of experiments, we investigate the effectiveness of our model in learning a classifier from incomplete multi-label data. We use the same datasets and experimental setup with the previous section, and compare with baseline methods for multi-label classification.

Methods Evaluated. Below we list all the methods evaluated in this set of experiment.

- *GT:* The proposed approach with ground-truth label hierarchy to learn the multi-label classifier.
- *learn:* The proposed method that simultaneously learns the tree structure and multi-label classifier.
- *BR:* This method is the binary relevance classifier without taking into account the label structure. That is, a logistic classifier is trained individually for each class.
- *tanCVPR [96]:* This is a relevant approach which learns a graph structure for the labels for multi-label classification. The difference with our approach is that it assumes complete labels and the learned structure is not necessarily a tree.
- *fasttag [24]:* This method aims to learn a multi-label classifier from incomplete labels. However, it does not take into account the structure of the labels.

Note that our method (*GT* and *learn*) and *BR* are based on logistic regression classifier, and *tanCVPR* is based on structure SVM classifier.

Performance Measures. We use two popular metrics to evaluate the performance of the learned multi-label classifiers, F1-Micro and F1-Macro. These two measures are based on the binary F1-measure that is calculated from the number of true positives (tp), true negatives (tn), false positives (fp) and false negatives (fn). In particular, for a class k, let tp_k, fp_k, tn_k and fn_k be the number of true positives, false positives, true negatives and false negatives, respectively. The F1-Micro and F1-macro are calculated

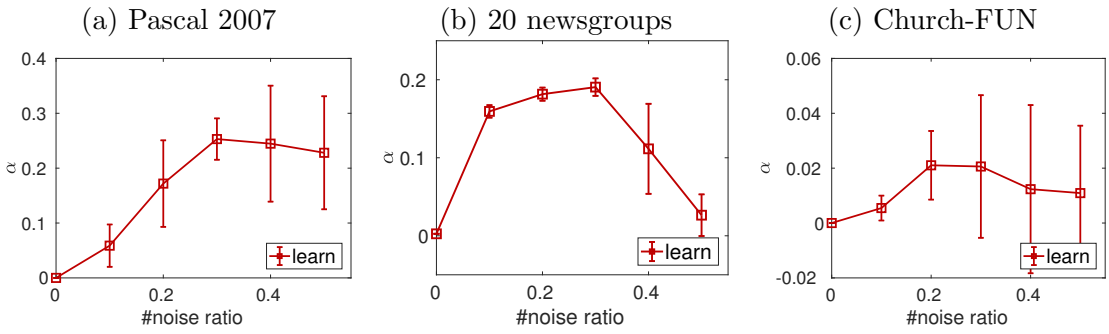


Figure 5.5: Estimated α value of the proposed method with different label noise ratios.

as follows:

$$\text{F1-Micro} = F1\Big(\sum_{k=1}^{K} tp_k, \sum_{k=1}^{K} fp_k, \sum_{k=1}^{K} tn_k, \sum_{k=1}^{K} fn_k\Big) ,$$

$$\text{F1-Macro} = \frac{1}{K}\sum_{k=1}^{K} F1(tp_k, fp_k, tn_k, fn_k) ,$$

where the base measure $F1(tp, fp, tn, fn) = \frac{2tp}{2tp+fp+fn}$. Note that when the classes are unbalanced, by definition, F1-macro would be more affected by the performance of the minority classes. In contrast, F1-micro would be more affected by the performance of the majority classes [97].

Results. We evaluate all the methods on the previous datasets using the same setting of noisy labels. Figure 5.4 shows the performance of the learned multi-label classifier for different methods.

First, we see that both *GT* and *learn* significantly outperform the BR method which does not make use of the label structure. This result demonstrates the benefits of employing the label structure for classification.

Secondly, we see that *GT* always performs better than *learn*, and outperforms other baselines in most cases. This result shows that, given the ground-truth label structure, our method is able to learn an effective multi-label classifier.

Thirdly, we see that without the ground-truth hierarchy, *learn* is able to learn a classifier that's competitive with *tanCVPR* and *fasttag*, which are state-of-the-art multi-label

classifiers. This demonstrates that in addition to learning a reasonable label hierarchy, our method can also simultaneously learn a reasonably good classifier. From the results, we observe that *fasttag* tends to perform better on classes with more examples while worse on those with fewer examples. This can be seen from the *Church-FUN* dataset, where its F1-Micro measure is generally lower than other methods and F1-Macro measure is generally higher. In contrast, our method is robust both in terms of the F1-Micro and the F1-Macro measures.

It is interesting to note that *tanCVPR* is also robust to the noise although it does not explicitly deal with missing labels. One possible reason is that *tanCVPR* uses an undirected graph to approximate the structure, which may provide more flexibility for learning the classifier. However, the graph structure learned by *tanCVPR* usually contains a lot of isolated nodes and the inferred graph structures are often drastically different from the ground truth hierarchy. In contrast, our method is capable of closely recovering the underlying hierarchy as shown in the previous section.

5.5.2.1 Further Investigation: Noise Estimation

Here we investigate how well our method can estimate the label noise in the incomplete data. We plot the values of the estimated noise factor α in Figure 5.5. We see that our method tends to underestimate the noise in the labels, especially when the noise rate is large. One possible reason is that due to the unknown structure of the labels, when the noise is large, the learning algorithm could simply change the structure to achieve higher objective, instead of increasing the estimation of α. In addition, the estimation of α is closely related to the variance of the learned classifier. We observe that when the noise rate is 0.4 or 0.5, the α is heavily underestimated, and the classification performance in Figure 5.4 also dropped accordingly. This suggests a future direction to improve the estimation of α so that the classification can also be further improved.

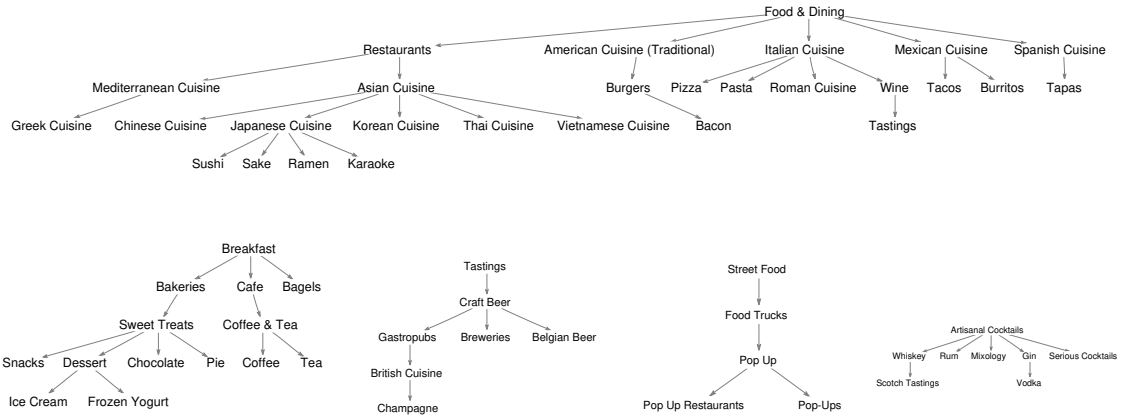


Figure 5.6: Subtrees of the learned label hierarchy on the Sosh data using the proposed method.

5.5.3 Case Study: Application to the Sosh Dataset

5.5.3.1 The Sosh Dataset

This dataset comes from Sosh, a previous technology company that focuses on tagging activities [99]. In this context, an activity is a single topic tagged by multiple resources. The labels in this dataset are all human-created and span everything from "Italian Cuisine" to "Vegan Friendly" to "National Burger Month". Due to the fact that the labels are created by multiple annotators without any careful coordination, it is natural that the label assignments are incomplete (assuming that the complete label set is the union of all the labels from all the resources). The original dataset contains 5126 activities with 491 different labels in total. Here we manually extracted 95 labels that are related to the broader "Food" category, and added a dummy "root" class. We remove the activities that is not positive for any of the 95 classes, resulting in 5053 data instances. We use the public Doc2Vec tool [61] to extract a 800-dimensional feature vector for each activity (400 for PV-DBOW features and 400 for PV-DM features as recommended in [61]). We then apply our method to learn the hierarchy from the data.

5.5.3.2 Results

Figure 5.6 shows subtrees of the learned label hierarchy on the Sosh dataset. We see that the structure contains some interesting hierarchical concepts. For example, "Food & Dining" is at the higher level of the hierarchy and contains some subcategories of cuisine such as "Restaurants", "Mexican Cuisine", "Italian Cuisine", etc. On a lower level, "Restaurants" has children "Asian Cuisine" and "Mediterranean Cuisine". In a more refined level, "Asian Cuisine" is the parent of "Chinese Cuisine", "Vietnamese Cuisine", "Korean Cuisine", and "Japanese Cuisine". Finally, "Japanese Cuisine" contains "Sushui", "Ramen", "Sake", and "Karaoke".

Such learned structure is meaningful, but may not be perfect. For example, the class "Wine" becomes the parent of "Tastings", which seems to be a reversed direction. This effect is possibly because that "Wine" could be the child of "Italian Cuisine" and "Tasting", while the former is more likely. Due to the restriction of a tree structure, "Italian Cuisine" is determined as the parent of "Wine", while "Tastings" becomes the child of "Wine". This implies that a more complex structure could be helpful, for example, a directed acyclic graph, which points out a future direction.

5.6 Related Work

There are three main lines of research that are related to our work. We list them below and describe some of the related work.

Hierarchical Multi-label Classification. This line of research focuses on Hierarchical Multi-label Classification (HMC), where the label hierarchy is assumed as a prior, and the goal is to learn the multi-label classifier that respects the hierarchy. HMC has been applied in several domains including protein function prediction [5, 9, 100, 118], text categorization [21, 54, 84, 126], music genre classification [19, 33, 98], image classification [11, 62], video annotation [35], and automatic classification of worldwide web documents [20, 81]. While these studies have shown that using label hierarchy is generally helpful for improving multi-label classifications, it usually requires significant domain knowledge to obtain the hierarchy and can be costly. In contrast, our work aims to simultaneously learn the classifier and discover the latent hierarchy, which is more desirable.

Graph Structure Learning for Multi-Label Data. There are two main cat-

egories of approaches that learn a graph structure from multi-label data. The first category separately learns the graph structure from labels, which can then be used to learn multi-label classifiers. Bradley and Guestrin [15] use the ChowLiu Tree algorithm [25] to construct a tree structure based on the mutual information of labels. In [65], the authors construct a maximum spanning tree using label co-occurrence. In [125], a directed acyclic graph is constructed to capture the joint probability of labels. In [94], the authors use an ensemble approach to constructs a graph from several random graphs. However, these graph constructions are often separated from the parameter learning process.

The second category simultaneously learns graph structure and the classifier [37, 88, 96, 108]. Ding et al. [37] study the joint learning of the graph structure and the parameters with a penalty term to encourage graph sparsity. Schmidt et al. [88] propose a framework to jointly learn pairwise CRFs and parameters with block-L1 regularization. Tan et al. [96] propose a clique generating method to learn graph structures for multi-label image classification. Wu et al. [108] propose a SVM-based approach to build the hierarchy of the multi-label of the data instances, instead of the labels. While these are interesting results, none of them can specifically deal with incomplete multi-label data.

Multi-label Learning with Incomplete Labels Assignments. This area of research studies multi-labeling learning with missing labels (e.g., [18, 24, 55]), and effective methods are proposed to deal with the label incompleteness. However, existing work generally considers a flat label structure. One related work is by Bi and Kwok [12], which takes into account the label correlation for learning. In contrast, we focus on the scenario where the labels form a hierarchy. Yu et al. [118] study hierarchical multi-label classification with missing labels on the leaf nodes, and propose an algorithm that considers both the hierarchical and the flat taxonomy similarity between labels to predict protein functions. Different from this work, in our problem, the hierarchy is unknown and the labels could be missing at the internal nodes of the hierarchy.

5.7 Conclusion

In this chapter, we studied the problem of learning with latent label hierarchy from incomplete multi-label data. Our goal was to simultaneously discover the label hierarchy and learn a multi-label classifier for the data with missing labels. We proposed a graph-

ical model to capture the label hierarchy and the generation process of the incomplete label assignments. Our objective was based on Maximum Likelihood Estimation. We developed an efficient EM optimization approach to estimate the classifier parameters and the hierarchy structure. Our results on real-world datasets demonstrated that our method can better discover the underlying hierarchy by utilizing both the labels and the input features, and learn a competitive multi-label classifier. Our application on a case study revealed interesting label hierarchy in the data with unknown structure.

Chapter 6: Conclusions and Future Work

6.1 Contributions

In this thesis, we studied four problems of learning with partial supervision for clustering and classification tasks.

First, we studied a novel instance clustering problem in the MIML framework, where the bag-level labels were used as side information to inform the clustering of instances. The goal was to find clusters that correspond to the classes or the subclasses within each class. We presented a simple yet effective principle that incorporates the bag-level label information as similarity constraints. The proposed constraints can be readily integrated into any optimization-based clustering algorithm by adding a penalty term to the objective. We demonstrated how the constraints can be incorporated into spectral clustering and empirically validated its effectiveness on both synthetic and real-world MIML datasets.

The second problem we addressed is clustering with instance-level similarity constraints. We proposed a unified framework for clustering with pairwise constraints and relative constraints, two common types of side information for clustering. We presented an extensive comparison between the two types of constraints by answering three research questions from two perspectives. From the user/annotator's perspective, we studied which type of constraint is easier to obtain from the human labelers, and which type of constraint allows for higher labeling accuracy. From the perspective of applying constraints to aid clustering, we compared the effectiveness of the two types of constraints in improving clustering. Our results revealed that pairwise constraints are generally easier to obtain, especially when a large portion of CL constraints exist. However, using the unified clustering model, the relative constraints were often more effective, even when the number of used relative constraints was far less than that of pairwise constraints, suggesting that relative constraints are more effective at improving clustering.

For the third problem, we studied active clustering, where the goal was to iteratively improve clustering by querying informative pairwise constraints. We introduced

a Bayesian clustering method that adopted a logistic clustering model and a data-dependent prior to control model complexity and encourage large cluster separations. Instead of directly computing the posterior of the clustering model at every iteration, our approach maintained a set of samples from the posterior. We presented a sequential MCMC method to efficiently update the posterior samples after obtaining a new pairwise constraint. We introduced two information-theoretic criteria to select the most informative pairs to query at every iteration. Experimental results demonstrated the effectiveness of the proposed Bayesian active clustering method over existing approaches.

Lastly, we studied the problem of learning with latent label hierarchy from incomplete multi-label data. Our goal was to simultaneously discover the label hierarchy and learn a multi-label classifier for the data with missing labels. We proposed a graphical model to capture the label hierarchy and the incomplete label assignments. Our objective was based on Maximum Likelihood Estimation, for which we developed an efficient EM optimization approach to estimate the classifier parameters and the hierarchy structure. Our results on real-world datasets demonstrated that our method can better discover the underlying hierarchy by utilizing both the labels and the input features, and learn a competitive multi-label classifier. Our application on a case study revealed interesting label hierarchy in the data with unknown label structure.

6.2 Future Work

In this work, we studied several problems for clustering and classification with partial supervision. Here we list some important future directions for this line of research.

- *Clustering with Hybrid Types of Constraints.* Our work showed that pairwise constraints are easier to obtain while they are not as effective as relative constraints for improving clustering. One future direction is to design methods to employ hybrid types of constraints in an efficient way.

- *Clustering with Richer Forms of Side Information.* Our work studied clustering with two types of side information, pairwise and relative constraints. In real applications, there could be other richer forms of user constraints, for example, comparative constraints involving a list of objects. One future direction is to introduce richer types of user inputs for clustering and design corresponding methods

for incorporating them into clustering.

- *Active Clustering with Hybrid or Richer Forms of Side Information.* Our work studied active learning for clustering with pairwise constraints. Different forms of partial information require different methods for active learning. It is important to design appropriate active learning methods for richer types of partial supervision.

- *Complex Label Structure for Multi-label Learning.* For our last problem of multi-label learning, we studied a problem where the labels from a hierarchy. There are many cases that the labels can form a more complex structure such as a directed acyclic graph or an undirected graph. Future work should consider learning such more complex structures to better capture the relationships between labels and improve classification.

Bibliography

[1] Muna Al-Razgan and Carlotta Domeniconi. Clustering ensembles with active constraints. In *Applications of Supervised and Unsupervised Ensemble Methods*, pages 175–189, 2009.

[2] Ehsan Amid, Aristides Gionis, and Antti Ukkonen. A kernel-learning approach to semi-supervised clustering with relative distance comparisons. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 219–234. Springer, 2015.

[3] Korinna Bade and Andreas Nürnberger. Hierarchical constraints. *Machine Learning*, 94(3):371–399, 2014.

[4] Mahdieh Soleymani Baghshah and Saeed Bagheri Shouraki. Semi-supervised metric learning using pairwise constraints. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 1217–1222, 2009.

[5] Zafer Barutcuoglu, Robert E Schapire, and Olga G Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, 2006.

[6] Sugato Basu, Arindam Banerjee, and Raymond J Mooney. Active semi-supervision for pairwise constrained clustering. In *SDM*, pages 333–344, 2004.

[7] Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 59–68, 2004.

[8] Michel Benaïm and Jean-Yves Le Boudec. On mean field convergence and stationary regime. *CoRR*, abs/1111.5710, 2011.

[9] Wei Bi and Jame T Kwok. Bayes-optimal hierarchical multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 27(11):2907–2918, 2015.

[10] Wei Bi and James T Kwok. Multi-label classification on tree-and dag-structured hierarchies. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 17–24, 2011.

[11] Wei Bi and James T Kwok. Hierarchical multilabel classification with minimum bayes risk. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 101–110. IEEE, 2012.

[12] Wei Bi and James T Kwok. Multilabel classification with label correlations and missing labels. In *AAAI*, pages 1680–1686, 2014.

[13] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21th International Conference on Machine Learning*, pages 81–88, 2004.

[14] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 1 edition, 2007.

[15] Joseph K Bradley and Carlos Guestrin. Learning tree conditional random fields. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 127–134, 2010.

[16] Forrest Briggs, Xiaoli Z. Fern, and Raviv Raich. Rank-loss Support Instance Machines for MIML Instance Annotation. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 534–542, New York, NY, USA, 2012. ACM.

[17] Forrest Briggs, Xiaoli Z. Fern, and Raviv Raich. Rank-loss support instance machines for miml instance annotation. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 534–542, 2012.

[18] Serhat Selcuk Bucak, Rong Jin, and Anil K Jain. Multi-label learning with incomplete class assignments. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2801–2808. IEEE, 2011.

[19] Juan José Burred and Alexander Lerch. A hierarchical approach to automatic musical genre classification. In *Proceedings of the 6th international conference on digital audio effects*, pages 8–11. Citeseer, 2003.

[20] Michelangelo Ceci and Donato Malerba. Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, 28(1):37–78, 2007.

[21] Soumen Chakrabarti, Byron Dom, Rakesh Agrawal, and Prabhakar Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *The VLDB Journal—The International Journal on Very Large Data Bases*, 7(3):163–178, 1998.

[22] Shiyu Chang, Charu C. Aggarwal, and Thomas S. Huang. Learning local semantic distances with limited supervision. In *Poceedings of the 14th IEEE International Conference on Data Mining*, pages 70–79, 2014.

[23] Shiyu Chang, Guo-Jun Qi, Charu C. Aggarwal, Jiayu Zhou, Meng Wang, and Thomas S. Huang. Factorized similarity learning in networks. In *Poceedings of the 14th IEEE International Conference on Data Mining*, pages 60–69, 2014.

[24] Minmin Chen, Alice Zheng, and Kilian Weinberger. Fast image tagging. In *International Conference on Machine Learning*, pages 1274–1282, 2013.

[25] C Chow and Cong Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.

[26] Yoeng-Jin Chu and Tseng-Hong Liu. On shortest arborescence of a directed graph. *Scientia Sinica*, 14(10):1396, 1965.

[27] Fan R K Chung. *Spectral Graph Theory*, volume 92. American Mathematical Society, 1997.

[28] Sanjoy Dasgupta. Analysis of a greedy active learning strategy. In *NIPS*, pages 337–344, 2005.

[29] Ian Davidson and S.S. Ravi. Identifying and Generating Easy Sets of Constraints For Clustering. In *Proceedings of the 21st AAAI conference on Artificial Intelligence*, pages 336–341. AAAI Press, 2006.

[30] Ian Davidson, Kiri Wagstaff, and Sugato Basu. Measuring Constraint-Set Utility for Partitional Clustering Algorithms. In *PKDD*, pages 115–126, 2006.

[31] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 209–216, 2007.

[32] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *ICML*, pages 209–216, 2007.

[33] Christopher DeCoro, Zafer Barutcuoglu, and Rebecca Fiebrink. Bayesian aggregation for hierarchical genre classification. In *ISMIR*, pages 77–80. Vienna, 2007.

[34] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 269–274. ACM, 2001.

[35] Anastasios Dimou, Grigorios Tsoumakas, Vasileios Mezaris, Ioannis Kompatsiaris, and Ioannis Vlahavas. An empirical study of multi-label learning methods for video annotation. In *Content-Based Multimedia Indexing, 2009. CBMI'09. Seventh International Workshop on*, pages 19–24. IEEE, 2009.

[36] Shifei Ding, Hongjie Jia, Liwen Zhang, and Fengxiang Jin. Research of semi-supervised spectral clustering algorithm based on pairwise constraints. *Neural Computing and Applications*, 24(1):211–219, 2014.

[37] Shilin Ding, Grace Wahba, and Xiaojin Zhu. Learning higher-order graph structure with features by structure penalty. In *Advances in Neural Information Processing Systems*, pages 253–261, 2011.

[38] Weisheng Dong, Xin Li, D Zhang, and Guangming Shi. Sparsity-based image denoising via dictionary learning and structural clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 457–464. IEEE, 2011.

[39] Jack Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240, 1967.

[40] Songhe Feng and De Xu. Transductive Multi-Instance Multi-Label learning algorithm with application to automatic image annotation. *Expert Syst. Appl.*, 37(1):661–670, January 2010.

[41] Charles W. Fox and Stephen J. Roberts. A tutorial on variational bayesian inference. *Artificial Intelligence Review*, 38(2):85–95, 2012.

[42] Walter R Gilks and Carlo Berzuini. Following a moving target – monte carlo inference for dynamic bayesian models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(1):127–146, 2001.

[43] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal bayesian active learning with noisy observations. In *NIPS*, pages 766–774, 2010.

[44] Ryan Gomes, Andreas Krause, and Pietro Perona. Discriminative clustering by regularized information maximization. In *Proceedings of Advances in Neural Information Processing Systems*, pages 775–783, 2010.

[45] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *NIPS*, pages 281–296, 2005.

[46] Derek Greene and Pádraig Cunningham. Constraint selection by committee: An ensemble approach to identifying informative constraints for semi-supervised clustering. In *ECML*, pages 140–151, 2007.

[47] Sandra G. Hart and Lowell E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Human Mental Workload*, volume 52 of *Advances in Psychology*, pages 139 – 183. North-Holland, 1988.

[48] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *CoRR*, abs/1112.5745, 2011.

[49] Kaizhu Huang, Yiming Ying, and Colin Campbell. Generalized sparse metric learning with relative comparisons. *Knowl. Inf. Syst.*, 28(1):25–45, 2011.

[50] Ruizhang Huang and Wai Lam. Semi-supervised document clustering via active learning with pairwise constraints. In *ICDM*, pages 517–522, 2007.

[51] Xiang Ji and Wei Xu. Document clustering with prior knowledge. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 405–412, New York, NY, USA, 2006. ACM.

[52] A Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Proceedings of Advances in Neural Information Processing Systems*, 14:841, 2002.

[53] Sepandar D. Kamvar, Dan Klein, and Christopher D. Manning. Spectral Learning. In *International Joint Conference on Artificial Intelligence, IJCAI*, pages 561–566, 2003.

[54] Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. Technical report, Stanford InfoLab, 1997.

[55] Xiangnan Kong, Zhaoming Wu, Li-Jia Li, Ruofei Zhang, Philip S Yu, Hang Wu, and Wei Fan. Large-scale multi-label learning with incomplete label assignments. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 920–928. SIAM, 2014.

[56] Frank R Kschischang, Brendan J Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory*, 47(2):498–519, 2001.

[57] Brian Kulis, Sugato Basu, Inderjit Dhillon, and Raymond Mooney. Semi-supervised Graph Clustering: a Kernel Approach. *Machine Learning Journal*, 74(1):1–22, January 2009.

[58] Nimit Kumar and Krishna Kummamuru. Semi-supervised clustering with metric learning using relative comparisons. *IEEE Trans. Knowl. Data Eng.*, 20(4):496–503, 2008.

[59] Tilman Lange, Martin H. C. Law, Anil K. Jain, and Joachim M. Buhmann. Learning with constrained and unlabelled data. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 731–738, 2005.

[60] Martin H. C. Law, Alexander P. Topchy, and Anil K. Jain. Model-based clustering with probabilistic constraints. In *Proc. SIAM Conf. on Data Mining*, pages 641–645, 2005.

[61] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

[62] Jurica Levatić, Dragi Kocev, and Sašo Džeroski. The importance of the label hierarchy in hierarchical multi-label classification. *Journal of Intelligent Information Systems*, 45(2):247–271, 2015.

[63] Tao Li and Mitsunori Ogihara. Detecting emotion in music, 2003.

[64] Weizhong Li, Lukasz Jaroszewski, and Adam Godzik. Clustering of highly homologous sequences to reduce the size of large protein databases. *Bioinformatics*, 17(3):282–283, 2001.

[65] Xin Li, Feipeng Zhao, and Yuhong Guo. Multi-label image classification with a probabilistic label enhancement model. In *UAI*, volume 1, page 3, 2014.

[66] Eric Yi Liu, Zhishan Guo, Xiang Zhang, Vladimir Jojic, and Wei Wang. Metric learning from relative comparisons by minimizing squared residual. In *Poceedings of the 12th IEEE International Conference on Data Mining*, pages 978–983, 2012.

[67] E.Y. Liu, Z. Zhang, and W. Wang. Clustering with relative constraints. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 947–955, 2011.

[68] Yi Liu, Rong Jin, and Anil K. Jain. Boostcluster: Boosting clustering by pairwise constraints. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 450–459, 2007.

[69] Yiyi Liu, Quanquan Gu, Jack P Hou, Jiawei Han, and Jian Ma. A network-assisted co-clustering algorithm to discover cancer subtypes based on gene expression. *BMC bioinformatics*, 15(1):37, 2014.

[70] Zhengdong Lu. Semi-supervised clustering with pairwise constraints: a discriminative approach. *Journal of Machine Learning Research*, 2:299–306, 2007.

[71] Zhengdong Lu and Todd K. Leen. Semi-supervised learning with penalized probabilistic clustering. In *Proceedings of Advances in Neural Information Processing Systems*, pages 849–856, 2004.

[72] Pavan Kumar Mallapragada, Rong Jin, and Anil K Jain. Active query selection for semi-supervised clustering. In *ICPR*, pages 1–4, 2008.

[73] MicrosoftResearch. Image database. `http://research.microsoft.com/en-us/projects/ObjectClassRecognition/`, 2005. [Online].

[74] Alexis Mignon and Frédéric Jurie. Pcca: A new approach for distance learning from sparse pairwise constraints. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2666–2672. IEEE, 2012.

[75] Radford M Neal. Slice sampling. *Annals of statistics*, 31(3):705–741, 2003.

[76] Blaine Nelson and Ira Cohen. Revisiting probabilistic models for clustering with pairwise constraints. In *Proceedings of the 24th International Conference on Machine Learning*, pages 673–680, 2007.

[77] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On Spectral Clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems, NIPS*, pages 849–856. MIT Press, 2001.

[78] M. E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, pages 722–729, Dec 2008.

[79] J.C. Nunnally and Ira Bernstein. *Psychometric Theory*. McGraw Hill, Inc., 1994.

[80] Yuanli Pei, Xiaoli Z. Fern, Rómer Rosales, and Teresa V. Tjahja. Discriminative clustering with relative constraints. *CoRR*, abs/1501.00037, 2014.

[81] Kunal Punera and Joydeep Ghosh. Enhanced hierarchical classification via isotonic smoothing. In *Proceedings of the 17th international conference on World Wide Web*, pages 151–160. ACM, 2008.

[82] David F Robinson and Leslie R Foulds. Comparison of phylogenetic trees. *Mathematical biosciences*, 53(1-2):131–147, 1981.

[83] Rómer Rosales and Glenn Fung. Learning sparse metrics via linear programming. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 367–373, 2006.

[84] Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. Learning hierarchical multi-category text classification models. In *Proceedings of the 22nd international conference on Machine learning*, pages 744–751. ACM, 2005.

[85] Juho Rousu, Craig Saunders, Sandor Szedmak, and John Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7(Jul):1601–1626, 2006.

[86] L. Saul, T. Jaakkola, and M. Jordan. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76, 1996.

[87] Mark Schmidt. L-bfgs software. Website, 2012. `http://www.di.ens.fr/~mschmidt/Software/minFunc.html`.

[88] Mark Schmidt and Kevin Murphy. Convex structure learning in log-linear models: Beyond pairwise potentials. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 709–716, 2010.

[89] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. In *Proceedings of Advances Neural Information Processing Systems*, pages 41–48, 2003.

[90] Noam Shental, Aharon Bar-Hillel, Tomer Hertz, and Daphna Weinshall. Computing gaussian mixture models with em using equivalence constraints. In *Proceedings of Advances in Neural Information Processing Systems*, pages 465–472, 2004.

[91] Jianbo Shi and Jitendra Malik. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, August 2000.

[92] Parag Singla and Pedro Domingos. Discriminative training of markov logic networks. In *AAAI*, volume 5, pages 868–873, 2005.

[93] Daniela Stojanova, Michelangelo Ceci, Donato Malerba, and Sašo Džeroski. Learning hierarchical multi-label classification trees from network data. In *International Conference on Discovery Science*, pages 233–248. Springer, 2013.

[94] Hongyu Su and Juho Rousu. Multilabel classification through random graph ensembles. In *ACML*, pages 404–418, 2013.

[95] Masashi Sugiyama. Dimensionality Reduction of Multimodal Labeled Data by Local Fisher Discriminant Analysis. *Journal of Machine Learning Research, JMLR*, 8:1027–1061, May 2007.

[96] Mingkui Tan, Qinfeng Shi, Anton van den Hengel, Chunhua Shen, Junbin Gao, Fuyuan Hu, and Zhen Zhang. Learning graph structure for multi-label image classification via clique generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4100–4109, 2015.

[97] Lei Tang, Suju Rajan, and Vijay K. Narayanan. Large scale multi-label classification via metalabeler. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 211–220, New York, NY, USA, 2009. ACM.

[98] Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis P Vlahavas. Multi-label classification of music into emotions. In *ISMIR*, volume 8, pages 325–330, 2008.

[99] Christopher Vanderschuere. *Improved text classification through label clustering.* PhD thesis, Oregon State University, 2015.

[100] Celine Vens, Jan Struyf, Leander Schietgat, Sašo Džeroski, and Hendrik Blockeel. Decision trees for hierarchical multi-label classification. *Machine Learning*, 73(2):185–214, 2008.

[101] Viet-Vu Vu, Nicolas Labroche, and Bernadette Bouchon-Meunier. An efficient active constraint selection algorithm for clustering. In *ICPR*, pages 2969–2972, 2010.

[102] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 577–584, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[103] Dingding Wang, Shenghuo Zhu, Tao Li, Yun Chi, and Yihong Gong. Integrating document clustering and multidocument summarization. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(3):14, 2011.

[104] Fei Wang, Chris H. Q. Ding, and Tao Li. Integrated KL (K-means - Laplacian) Clustering: A New Clustering Approach by Combining Attribute Data and Pairwise Relations. In *SIAM SDM*, pages 38–48, 2009.

[105] Xiang Wang and Ian Davidson. Active spectral clustering. In *ICDM*, pages 561–568, 2010.

[106] Xiang Wang and Ian Davidson. Flexible constrained spectral clustering. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, KDD '10, pages 563–572, New York, NY, USA, 2010. ACM.

[107] Xiang Wang, Jun Wang, Buyue Qian, Fei Wang, and Ian Davidson. Self-taught spectral clustering via constraint augmentation. In *Proc. SIAM Conf. on Data Mining*, pages 416–424, 2014.

[108] Qingyao Wu, Yunming Ye, Haijun Zhang, Tommy WS Chow, and Shen-Shyang Ho. Ml-tree: A tree-structure-based approach to multilabel learning. *IEEE transactions on neural networks and learning systems*, 26(3):430–443, 2015.

[109] Shiming Xiang, Feiping Nie, and Changshui Zhang. Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition*, 41(12):3600 – 3612, 2008.

[110] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart J. Russell. Distance metric learning with application to clustering with side-information. In *Proceedings of Advances Neural Information Processing Systems*, pages 505–512, 2002.

[111] Sicheng Xiong, Javad Azimi, and Xiaoli Z Fern. Active learning of constraints for semi-supervised clustering. *IEEE Trans. Knowl. Data Eng.*, 26(1):43–54, 2014.

[112] Qianjun Xu, Marie desJardins, and Kiri L Wagstaff. Active constrained clustering by examining spectral eigenvectors. In *Discovery Science*, pages 294–307. Springer, 2005.

[113] Xin-Shun Xu, Xiangyang Xue, and Zhi-Hua Zhou. Ensemble Multi-Mnstance Multi-Label Learning Approach for Video Annotation Task. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 1153–1156, New York, NY, USA, 2011. ACM.

[114] Xiangyang Xue, Wei Zhang, Jie Zhang, Bin Wu, Jianping Fan, and Yao Lu. Correlative multi-label multi-instance image annotation. In *IEEE International Conference on Computer Vision*, pages 651–658. IEEE, 2011.

[115] Liu Yang, Rong Jin, and Rahul Sukthankar. Bayesian active distance metric learning. In *UAI*, pages 442–449, 2007.

[116] Shuang-Hong Yang, Jiang Bian, and Hongyuan Zha. Hybrid Generative/Discriminative Learning for Automatic Image Annotation. In *Conference on Uncertainty in Artificial Intelligence, UAI*, pages 683–690. AUAI Press, 2010.

[117] Xuesong Yin, Songcan Chen, Enliang Hu, and Daoqiang Zhang. Semi-supervised clustering with metric learning: an adaptive kernel method. *Pattern Recognition*, 43(4):1320–1333, 2010.

[118] Guoxian Yu, Hailong Zhu, and Carlotta Domeniconi. Predicting protein functions using incomplete hierarchical labels. *BMC Bioinformatics*, 16(1):1, 2015.

[119] Stella X. Yu and Jianbo Shi. Grouping with bias. In *Advances in Neural Information Processing Systems, NIPS*, pages 1327–1334, Cambridge, MA, 2001. MIT Press.

[120] Stella X. Yu and Jianbo Shi. Multiclass spectral clustering. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, ICCV '03, pages 313–319, Washington, DC, USA, 2003. IEEE Computer Society.

[121] Stella X. Yu and Jianbo Shi. Segmentation given partial grouping constraints. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):173–183, January 2004.

[122] Lihi Zelnik-Manor and Pietro Perona. Self-Tuning Spectral Clustering. In *Advances in Neural Information Processing Systems, NIPS*, pages 1601–1608, Cambridge, MA, USA, 2004. MIT Press.

[123] Hong Zeng and Yiu-ming Cheung. Semi-supervised maximum margin clustering with pairwise constraints. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 24(5):926–939, 2012.

[124] Zheng-Jun Zha, Xian-Sheng Hua, Tao Mei, Jingdong Wang, Guo-Jun Qi, and Zengfu Wang. Joint Multi-Label Multi-Instance Learning for Image Classification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1–8. IEEE Computer Society, June 2008.

[125] Min-Ling Zhang and Kun Zhang. Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 999–1008. ACM, 2010.

[126] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.

[127] Min-Ling Zhang and Zhi-Hua Zhou. M^3MIML: A Maximum Margin Method for Multi-Instance Multi-Label Learning. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 688–697, Washington, DC, USA, December 2008. IEEE Computer Society.

[128] Xiangrong Zhang, Licheng Jiao, Fang Liu, Liefeng Bo, and Maoguo Gong. Spectral clustering ensemble applied to sar image segmentation. *IEEE Transactions on Geoscience and Remote Sensing*, 46(7):2126–2136, 2008.

[129] Zhi-Hua Zhou and Min-Ling Zhang. Multi-Instance Multi-Label Learning with Application to Scene Classification. In *Advances in Neural Information Processing Systems, NIPS*, pages 1609–1616. MIT Press, Cambridge, MA, USA, 2007.

[130] Zhi-Hua Zhou and Min-Ling Zhang. Multi-instance multi-label learning with application to scene classification. *Advances in neural information processing systems*, 19:1609, 2007.

APPENDICES

Appendix A: Appendix for Manuscript 1

Appendix

In this appendix, we show the derivation from Eq. (2.10) to Eq. (2.11). First we focus on the summation term inside the trace operation. Since the two summation sum over all possible configurations of (i,j) and (r,s), we have the following rearrangement

$$\begin{aligned}
& \sum_{(i,j)}\sum_{(r,s)}(y_i^T y_j - y_r^T y_s)(b_j b_i^T - b_s b_r^T) \\
= & \sum_{(i,j)}\sum_{(r,s)}\left[(y_i^T y_j)b_j b_i^T + (y_r^T y_s)b_s b_r^T - (y_r^T y_s)b_j b_i^T - (y_i^T y_j)b_s b_r^T\right] \\
= & 2[\sum_{(i,j)}\sum_{(r,s)}(y_i^T y_j)b_j b_i^T - \sum_{(i,j)}\sum_{(r,s)}(y_i^T y_j)b_s b_r^T] \\
= & 2[\sum_{(i,j)} b_j(y_i^T y_j)b_i^T \sum_{(r,s)} 1 - \sum_{(i,j)}(y_i^T y_j)\sum_{(r,s)} b_s b_r^T] \\
= & 2[M^2 \sum_{(i,j)} b_j(y_i^T y_j)b_i^T - \mathbf{1}^\top (Y^T Y)\mathbf{1} B B^T] \\
= & 2M^2 \cdot [\sum_{(i,j)} b_j(y_j^T y_i)b_i^T - \frac{\mathbf{1}^\top (Y^T Y)\mathbf{1}}{M^2} B B^T] \\
= & 2M^2 \cdot (B Y^T Y B^T - \frac{\mathbf{1}^\top (Y^T Y)\mathbf{1}}{M^2} B B^T) \\
= & 2M^2 \cdot B(Y^T Y - \mu I)B^T
\end{aligned}$$

where $\mu = \frac{\mathbf{1}^\top (Y^T Y)\mathbf{1}}{M^2}$, and I is the identity matrix. In the derivation, we have used the fact that $y_i^T y_j$ is a scalar and $y_i^T y_j = y_j^T y_i$. Correspondingly, Eq. (2.10) is derived to Eq. (2.11) as

$$\begin{aligned}
& \frac{\alpha}{2M^2} \cdot tr\left(Z^T D^{-1/2}[2M^2 B(Y^T Y - \mu I)B^T]D^{-1/2} Z\right) \\
= & \alpha \cdot tr\left(Z^T D^{-1/2} B(Y^T Y - \mu I)B^T D^{-1/2} Z\right).
\end{aligned}$$