Modeling Big Data Variety with Graph Mining Techniques

BY

XIANGNAN KONG
B.S., Nanjing University, 2006
M.S., Nanjing University, 2009

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2014

Chicago, Illinois

Defense Committee:

Philip S. Yu, Chair and Advisor
Bing Liu
John Lillis
Junhui Wang, Statistics
Ann B. Ragin, Northwestern University

This thesis is dedicated to my parents and my wife

for their love, endless support,

and encouragement.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Philip S. Yu. I cannot thank him enough for introducing me to the exciting world of research. He certainly has changed my life. Working with Philip has always been a delightful, rewarding experience. He provided a research environment in which I could freely explore different interesting topics. During my PhD years at UIC, Philip has been very supportive, giving me invaluable advices from many aspects: instruction on research, writing, insightful guidance on academia, mentoring students, collaboration with other researchers. Without his great help, support, and encouragement, I would certainly not be where I am today. He is more than just a great advisor, he is a friend, and that combination has made the past five years seem to pass very quickly. Its hard to believe its coming to an end.

I would also like to thank all my collaborators and committiee members. Their comments and encouragements helped me to tackle many difficulties in research.

I owe an enormous amount to my family who sacrificed a lot during my student years. To my father, who is alway supportive for my decision to pursuit of Master's degree and PhD degree. To my mother, who always loved me even when she was suffering from depression during her last years. My passion on drug discovery research and neuroscience is fostered by her love. To my wife, Yang, who willingly endured all my hours in front of the computer, for being my joy and my love.

XK

ACKNOWLEDGMENTS (Continued)

Portions of chapters 2, 3, 4, 5 in this work have been published previously in (1; 2; 3; 4) .

Contribution of Authors

Chapter 1 is an introduction that outlines my dissertation research. Chapter 2 represents a published manuscript (1) for which I was the primary author and major driver of the research. My research advisor, Dr. Philip S. Yu contributed to the writing of the manuscript. Chapter 3 represents a published manuscript (2) for which I was the primary author. My research advisor, Dr. Philip S. Yu contributed to the writing of the manuscript. Chapter 4 represents a published manuscript (3) for which I was the primary author. Dr. Wei Fan contributed by discussing the problem and ideas with me. My research advisor, Dr. Philip S. Yu contributed to the writing of the manuscript. In Chapter 5 represents a published manuscript (4) for which I was the primary author. My research advisor, Dr. Philip S. Yu contributed to the writing of the manuscript. Dr. Ann B. Ragin contributed the HIV data collection in the experiment. Dr. Xue Wang helped me to set up data preprocessing pipeline.

TABLE OF CONTENTS

CHAPTER			PAGE
1	**INTRODUCTION**		1
	1.1	Thesis Outline	1
	1.2	Multi-label Graph Classification	3
	1.3	Semi-supervised Graph Classification	4
	1.4	Dual Active Feature and Sample Selection for Graph Classification	4
	1.5	Uncertain Graph Classification for Brain Networks	5
2	**MULTI-LABEL GRAPH CLASSIFICATION**		7
	2.1	Introduction	7
	2.2	Related Work	12
	2.3	Problem Formulation	14
	2.4	Optimization Framework	16
	2.5	The Proposed Solution	21
	2.5.1	Subgraph Enumeration	22
	2.5.2	Upper Bound of gHSIC	23
	2.5.3	Subgraph Search Space Pruning	24
	2.6	Exploiting Label Correlations	26
	2.7	Experiments	29
	2.7.1	Experimental Setup	29
	2.7.1.1	Data Collections	29
	2.7.1.2	Evaluation Metrics	30
	2.7.1.3	Comparing Methods	33
	2.7.2	Performances on Multi-label Graph Classification	35
	2.7.3	Effectiveness of Subgraph Search Space Pruning	41
	2.7.4	Effectiveness of Embedding Label Correlations	42
	2.8	Conclusion	44
3	**SEMI-SUPERVISED GRAPH CLASSIFICATION**		45
	3.1	Introduction	45
	3.2	Related Work	49
	3.3	Problem Formulation	50
	3.3.1	Semi-Supervised Feature Selection	50
	3.3.2	Optimization Framework	52
	3.4	gSSC	56
	3.4.1	Subgraph Mining	57
	3.4.2	Upper Bound of gSemi	58

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

3.4.3 Pruning Search Space . . . 59
3.4.4 Discussion . . . 61
3.5 Experiments . . . 63
3.5.1 Experimental Setup . . . 63
3.5.2 Performances on Graph Classification . . . 66
3.5.3 Pruning Search Space . . . 67
3.5.4 Parameter Settings . . . 70
3.6 Conclusion . . . 71

4 DUAL ACTIVE FEATURE AND SAMPLE SELECTION FOR GRAPH CLASSIFICATION . . . 72
4.1 Introduction . . . 72
4.2 Related Work . . . 77
4.3 Problem Formulation . . . 77
4.3.1 Optimization Framework . . . 80
4.3.1.1 The Intuition . . . 81
4.3.1.2 The Solution . . . 82
4.3.1.3 Upper Bound of gFScore . . . 86
4.4 Experiments . . . 87
4.4.1 Performance on Graph Classification . . . 91
4.4.2 Parameter Settings . . . 92
4.5 Conclusions . . . 93

5 UNCERTAIN GRAPH CLASSIFICATION FOR BRAIN NETWORKS . . . 96
5.1 Introduction . . . 96
5.2 Related Work . . . 101
5.3 Problem Formulation . . . 102
5.4 The Proposed Framework . . . 106
5.4.1 Discrimination Score Distribution . . . 106
5.4.2 Efficient Computation . . . 109
5.4.3 Upper-Bounds for Subgraph Pruning . . . 114
5.5 Experiments . . . 115
5.5.1 Data Collection . . . 116
5.5.2 Comparative Methods . . . 117
5.5.3 Performance on Uncertain Graph Classification . . . 120
5.5.4 Influence of Parameter . . . 123
5.6 Conclusion . . . 124

6 CONCLUSION . . . 125

APPENDICES . . . 128

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

CITED LITERATURE 130

VITA 137

LIST OF TABLES

TABLE		PAGE
I	Summary of experimental tasks studied. "AvgL" denotes the average number of labels assigned to each graph.	28
II	Details of the anti-cancer activity prediction task (NCI1 dataset). Each label represents the assay result for one type of cancer. "Pos (%)" denotes the average percentage of positive instances for each cancer assay.	28
III	Details of toxicology prediction task (PTC dataset), where each of the multiple labels represents the toxicology test result on one type of animal. "Pos (%)" denotes the average percentage of positive instances for each cancer assay.	28
IV	Details of kinase inhibition prediction task (NCI2 dataset), where each of the multiple labels represents the inhibition of one type of kinase. "Pos (%)" denotes the average percentage of positive instances for each cancer assay.	28
V	SUMMARY OF EXPERIMENTAL DATASETS. "POS%" DENOTES THE AVERAGE PERCENTAGE OF POSTIVE GRAPHS IN EACH DATASET.	65
VI	Important Notations.	78
VII	SUMMARY OF EXPERIMENTAL DATASETS. "# POS" DENOTES THE NUMBER OF ACTIVE GRAPHS IN THE DATASET.	89
VIII	IMPORTANT NOTATIONS.	100
IX	SUMMARY OF DISCRIMINATION SCORE FUNCTIONS.	110
X	SUMMARY OF EXPERIMENTAL DATASETS.	118
XI	Results on the ADNI (Alzheimer's Disease) dataset with different number of features($t = 100, \cdots, 500$). The results are reported as "average performance + (rank)".	119

LIST OF TABLES (Continued)

TABLE **PAGE**

XII Results on the ADHD (Attention Deficit Hyperactivity Disorder) dataset with different number of features ($t = 100, \cdots, 500$). The results are reported as "average performance + (rank)". 120

XIII Results on the HIV (Human Immunodeficiency Virus) dataset with different number of features ($t = 100, \cdots, 500$). The results are reported as "average performance + (rank)". 121

LIST OF FIGURES

FIGURE		PAGE
1	Big data: expending in three fronts.	2
2	Examples of multi-label graphs. a) In kinase inhibition, each molecule can inhibit the activities of multiple types of kinases; b) In anti-cancer prediction, each molecular medicine can have anti-cancer efficacies on multiple types of cancers; c) In toxicology analysis, each chemical compound has carcinogenicity activities in multiple animal models.	8
3	Two types of Feature Selection Processes for Multi-label Graph Classification	9
4	The gMLC algorithm	25
5	Multi-label graph classification performances on Anti-cancer Activity Prediction (NCI1 dataset)	35
6	Multi-label graph classification performances on Kinase Inhibition Prediction (NCI2 dataset)	36
7	Multi-label graph classification performances on Toxicology Prediction Task (PTC dataset)	37
8	Average CPU time for nested gMLC versus un-nested gMLC with varying min_sup.	38
9	Average number subgraph patterns explored during mining for nested gMLC versus un-nested gMLC with varying min_sup.	38
10	Performances of gMLC with/without considering label correlations on anti-cancer activity prediction task (NCI1 dataset)	43
11	An example of semi-supervised feature selection on graph data. The subgraph feature "a-b" is more useful than "a-c" based on both labeled and unlabeled graphs.	46
12	Different Feature Selection Frameworks for Graph Classification	49

LIST OF FIGURES (Continued)

FIGURE **PAGE**

13 The gSSC algorithm 60

14 Classification accuracy with different number of features. (#label=30) 62

15 Classification accuracy with different number of features. (#label=50) 63

16 Classification accuracy with different number of features. (#label=70) 64

17 Average CPU time for nested gSSC versus un-nested gSSC with varying min_sup. 68

18 Average number subgraph patterns explored during mining for nested gSSC versus un-nested gSSC with varying min_sup. 68

19 Classification accuracy of gSSC with different $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. (#label=50) . 69

20 An example of dual active feature and sample selection in graph data. . 73

21 Different active learning frameworks for graph classification. 76

22 Graph distances using two optimal feature sets ($\mathcal{T}_s^+$ and $\mathcal{T}_s^-$) respectively depending on the label of the query graph. 80

23 The gActive algorithm 84

24 Candidate subgraph pattern lists 85

25 Graph classification accuracy after different number of graphs being queried. 94

26 gActive accuracies with different α and β. 95

27 CPU time with/without pruning. 95

28 An example of uncertain graph classification task. 98

29 Different types of subgraph features for uncertain graph classification . 99

30 The dynamic programming process for computing $\Pr\left[n_+^g, \widetilde{\mathcal{D}}_+\right]$. 112

31 The dynamic programming algorithm for probability computation. . . . 113

LIST OF FIGURES (Continued)

FIGURE **PAGE**

32 Recursive equation for dynamic programming. 114

33 The Dug framework for discriminative subgraph mining. 114

34 Parameter Studies (ADNI dataset). 123

SUMMARY

Graphs are ubiquitous and have become increasingly important in modeling diverse kinds of objects. In many real-world applications, instances are not represented as feature vectors, but as graphs with complex structures, *e.g.*, chemical compounds, program flows, XML web documents and brain networks. One central issue in graph mining research is graph classification, which has a wide variety of real world applications, *e.g.* drug activity predictions, toxicology tests and kinase inhibitions.

There are some major challenges in real-world graph classification problems as follows:

•*Learning from graphs with multiple labels:* For example, in Figure 2, a chemical compound can inhibit the activities of multiple types of kinases, *e.g.*, *ATPase* and *MEK kinase*; One drug molecular can have anti-cancer efficacies on multiple types of cancers.

•*Learning from a small number of labeled graphs:* In many real world applications, the labels of graph data are very expensive or difficult to obtain. Creating a large training dataset can be too expensive, time-consuming or even infeasible. For example, in molecular medicine, it requires time, efforts and excessive resources to test drugs' anti-cancer efficacies by pre-clinical studies and clinical trials, while there are often copious amounts of unlabeled drugs or molecules available from various sources.

•*Learning from uncertain graphs:* For example, in neuroimaging, the functional connectivities among different brain regions are highly uncertain (5; 6; 7; 8). In such applications, each human brain can be represented as an uncertain graph, instead of a certain graph.

SUMMARY (Continued)

In this thesis, we explore four different settings of graph classification: multi-label setting, semi-supervised setting, active learning setting, and uncertain graph setting. In the multi-label setting, each graph object can be assigned with multiple labels. In semi-supervise setting and active learning setting, we explore two different settings to reduce the labeling costs in graph classification problems. In uncertain graph setting, we explore how to incorporate the uncertainty information in the graph structure for graph classification problems.

CHAPTER 1

INTRODUCTION

1.1 Thesis Outline

The objective of my thesis research is to design computational systems that are capable of analyzing and processing big data. The term "big data" usually refers to 3V's, *i.e.*, Volume (the amount of data), Velocity (the speed of data generated), and Variety (the kinds of data available). My thesis research mainly focuses on addressing the variety issues of big data. *Data variety* is about the increasing number of data types that need to be handled differently from simple logs and conventional data records, and also many different data sources that need to be fused together. These include data collected from scientific studies, health care records, social networks and social media: user activities, events, locations, *etc.* Although technology advances (*e.g.*, Apache Hadoop) have helped us enormously in dealing with the first two V's (volume and velocity), data variety remains a challenging problem to solve programmatically, and once it succeeds, it will represent a fundamental advance in big data research with great benefits to diverse fields, such as biomedical research, social computing, business, *etc.*

The data variety issues are difficult to solve because the data usually have complex structures and involve many different types of information interrelated in a complex way. For example, in computational system biology, if we want to predict the efficacy of a molecular drug for a certain disease, there are many challenging issues we need to address. Chemical compounds have

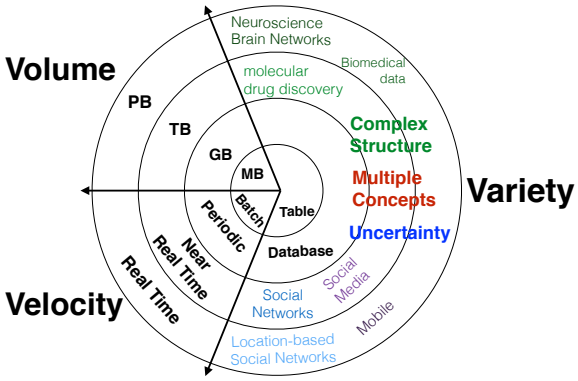


Figure 1. Big data: expending in three fronts.

complex molecular structures that standard computational methods cannot handle directly. The biology system is also extremely complex, involving multiple concepts, such as different genes, pathways, tissues, that are interrelated. In computational neuroscience, if we want to analyze the connectivity pattern among different regions of the human brian, we also need to address many challenging issues, such as the uncertainty in the linkage structure between brain regions.

A successful data model for data variety must be able to formulate the complex structures and multiple types of information of the data in a holistic perspective. Graphs are becoming increasingly important in modeling real-world data with complex structures. Examples include chemical compounds, brain networks and social networks. These data are quite different from traditional data objects, which have flat features and thus can be processed easily using conventional methods. A graph object (*e.g.*, a chemical compound) usually has a complex structure, represented as a set of nodes (*e.g.*, atoms) interconnected through a set of edges (*e.g.*, chemical bonds). There is no feature readily available.

In the following sections, I will outline our work on modeling various structural data as graph objects, and extracting important subgraph patterns in many different scenarios: drug discovery tasks where the labeling costs are very high or when we want to explore multiple gene targets; neuroimaging tasks where the structures of brain networks are highly noisy and uncertain. In such scenarios, significant, discriminative and reliable patterns for structural data would be of great utility and the prerequisite for any true analytics.

1.2 Multi-label Graph Classification

Graph classification has been showing critical importance in a wide variety of applications, *e.g.* drug activity predictions and toxicology analysis. Current research on graph classification focuses on single-label settings. However, in many applications, each graph data can be assigned with a set of multiple labels simultaneously. Extracting good features using multiple labels of the graphs becomes an important step before graph classification. In Chapter 2, we study the problem of multi-label feature selection for graph classification and propose a novel solution, called gMLC, to efficiently search for optimal subgraph features for graph objects with multiple labels. Different from existing feature selection methods in vector spaces which assume the feature set is given, we perform multi-label feature selection for graph data in a progressive way together with the subgraph feature mining process. We derive an evaluation criterion to estimate the dependence between subgraph features and multiple labels of graphs. Then a branch-and-bound algorithm is proposed to efficiently search for optimal subgraph features by judiciously pruning the subgraph search space using multiple labels. Empirical studies demon-

strate that our feature selection approach can effectively boost multi-label graph classification performances and is more efficient by pruning the subgraph search space using multiple labels.

1.3 Semi-supervised Graph Classification

Current research on graph classification also assumes the existence of large amounts of labeled training graphs. However, in many applications, the labels of graph data are very expensive or difficult to obtain, while there are often copious amounts of unlabeled graph data available. In Chapter 3, we study the problem of semi-supervised feature selection for graph classification and propose a novel solution, called gSSC, to efficiently search for optimal subgraph features with labeled and unlabeled graphs. Different from existing feature selection methods in vector spaces which assume the feature set is given, we perform semi-supervised feature selection for graph data in a progressive way together with the subgraph feature mining process. We derive a feature evaluation criterion, named gSemi, to estimate the usefulness of subgraph features based upon both labeled and unlabeled graphs. Then we propose a branch-and-bound algorithm to efficiently search for optimal subgraph features by judiciously pruning the subgraph search space. Empirical studies on several real-world tasks demonstrate that our semi-supervised feature selection approach can effectively boost graph classification performances with semi-supervised feature selection and is very efficient by pruning the subgraph search space using both labeled and unlabeled graphs.

1.4 Dual Active Feature and Sample Selection for Graph Classification

Current research on graph classification focuses on mining discriminative subgraph features under supervised settings. The basic assumption is that a large number of labeled graphs are

available. However, labeling graph data is quite expensive and time consuming for many real-world applications. In order to reduce the labeling cost for graph data, we address the problem of how to select the most important graph to query for the label. This problem is challenging and different from conventional active learning problems because there is no predefined feature vector. Moreover, the subgraph enumeration problem is NP-hard. The active sample selection problem and the feature selection problem are *correlated* for graph data. Before we can solve the active sample selection problem, we need to find a set of optimal subgraph features. To address this challenge, in Chapter 4, we demonstrate how one can simultaneously estimate the usefulness of a query graph and a set of subgraph features. The idea is to maximize the dependency between subgraph features and graph labels using an active learning framework. We propose a branch-and-bound algorithm to search for the optimal query graph and optimal features simultaneously. Empirical studies on nine real-world tasks demonstrate that the proposed method can obtain better accuracy on graph data than alternative approaches.

1.5 Uncertain Graph Classification for Brain Networks

Mining discriminative features for graph data has attracted much attention in recent years due to its important role in constructing graph classifiers, generating graph indices, etc. Most measurement of interestingness of discriminative subgraph features are defined on certain graphs, where the structure of graph objects are certain, and the binary edges within each graph represent the "presence" of linkages among the nodes. In many real-world applications, however, the linkage structure of the graphs is inherently uncertain. Therefore, existing measurements of interestingness based upon certain graphs are unable to capture the structural uncertainty in

these applications effectively. In Chapter 5, we study the problem of discriminative subgraph feature selection from uncertain graphs. This problem is challenging and different from conventional subgraph mining problems because both the structure of the graph objects and the discrimination score of each subgraph feature are uncertain. To address these challenges, we propose a novel discriminative subgraph feature selection method, DUG, which can find discriminative subgraph features in uncertain graphs based upon different statistical measures including expectation, median, mode and φ-probability. We first compute the probability distribution of the discrimination scores for each subgraph feature based on dynamic programming. Then a branch-and-bound algorithm is proposed to search for discriminative subgraphs efficiently. Extensive experiments on various neuroimaging applications (*i.e.*, Alzheimer's Disease, ADHD and HIV) have been performed to analyze the gain in performance by taking into account structural uncertainties in identifying discriminative subgraph features for graph classification.

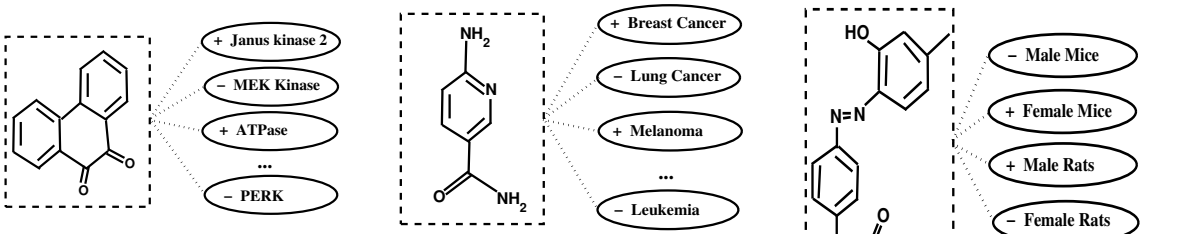
CHAPTER 2

MULTI-LABEL GRAPH CLASSIFICATION

2.1 Introduction

Due to the recent advances of data collection technology, many application fields are facing various data with complex structures, *e.g.*, chemical compounds, program flows and XML web documents. Different from traditional data in feature spaces, these data are not represented as feature vectors, but as graphs which raise one fundamental challenge for data mining research: the complex structure and lack of vector representations (9; 10; 11; 12). An effective model for graph data should be able to extract or find a proper set of features for these graphs in order to perform analysis or management steps. Motivated by these challenges, graph mining research problems, in particular graph classification, have received considerable attention in the last decade.

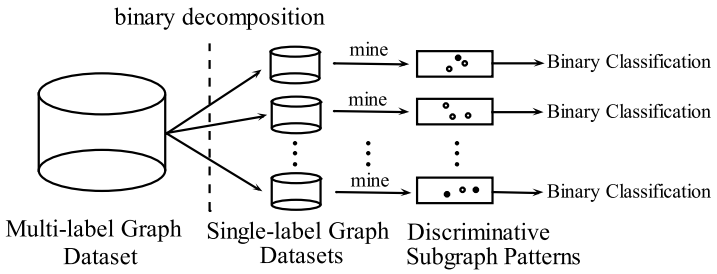
In the literature, graph classification problem has been extensively studied. Conventional approaches focus on single-label classification problems (13; 14; 15; 16), which assume, explicitly or implicitly, that each graph has only one label. However, in many real-world applications, each graph can be assigned with more than one label. For example, in Figure 2, a chemical compound can inhibit the activities of multiple types of kinases, *e.g.*, *ATPase* and *MEK kinase*; One drug molecular can have anti-cancer efficacies on multiple types of cancers. The selection and discovery of drugs or kinase inhibitors can be significantly improved if these chemical molecules



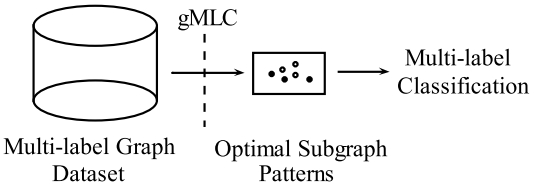
(a) Kinase Inhibitor (CID = 6763) (b) Anti-Cancer Drug (CID = 9500) (c) Toxicology Analysis (ID = TR222)

Figure 2. Examples of multi-label graphs. a) In kinase inhibition, each molecule can inhibit the activities of multiple types of kinases; b) In anti-cancer prediction, each molecular medicine can have anti-cancer efficacies on multiple types of cancers; c) In toxicology analysis, each chemical compound has carcinogenicity activities in multiple animal models.

are automatically tagged with a set of multiple labels or potential efficacies. This setting is also known as multi-label classification where each instance can be associated with multiple categories. It has been shown useful in many real-world applications such as text categorization (17; 18) and bioinformatics (19). Multi-label classification is particularly challenging on graph data. The reason is that, in the single-label case, conventional graph mining methods can extract or find one set of discriminative subgraph features for the single label concept within the graph dataset. But in multi-label cases, each graph contains multiple label concepts, and multiple sets of subgraph features should be mined, one for each label concept, in order to decide all the possible categories for each graph using binary classifiers (one-vs-all technique (20)). Thus the time and memory used for classifying multi-label graph data is much larger than for the single-label graphs. A major difficulty in performing multi-label classification on



(a) One-vs-All Single-label Feature Selection Process for Multi-label Graph Classification



(b) gMLC Feature Selection Process for Multi-label Graph Classification

Figure 3. Two types of Feature Selection Processes for Multi-label Graph Classification

graph data lies in the complex structure of graphs and lack of features which is useful for multiple labels concepts. Selecting a proper set of features for graph data becomes an essential and important procedure for multi-label graph classification.

Despite its value and significance, the multi-label feature selection problem for graph data has not been studied in this context so far. If we consider graph mining and multi-label classification as a whole, the major research challenges on multi-label feature selection for graph classification are as follows:

1. **Graph Data:** One fundamental problem in multi-label feature selection on graph data lies in the complex structures and lack of feature representations of graphs. Conventional feature selection approaches in vector spaces assume, explicitly or implicitly, that a full set

of features is given before the feature selection. In the context of graph data, however, the full set of features for a graph dataset, are usually too large or even infeasible to obtain. For example, in graph mining, the number of subgraph features grows exponentially with the size of the graphs, which makes it impossible to enumerate all the subgraph features before the feature selection.

2. **Multiple Labels:** Another fundamental problem in multi-label feature selection on graph data lies in the multiple label concepts for each graph, *i.e.* how to utilize the multiple label concepts in a graph dataset to find a proper set of subgraph features for classification tasks. Conventional feature selection in graph classification approaches focuses on single-labeled settings (21; 13; 14). The mining strategy of discriminative subgraph patterns strictly follows the assumption that each graph has only one label. However, in many real-world applications, one graph can usually be assigned with multiple labels simultaneously. Directly applying single-label graph feature selection methods by adopting the popular one-versus-all binary decomposition (3(a)), which performs feature selection on each label concept, will result in different sets of subgraph features on different classes. Thus most state-of-the-art multi-label classification approaches in vector spaces cannot be used, since they assume that the instances should have a same set of features in the input space (18; 19).

3. **Label Correlations:** In many real-world applications, the multiple labels of graphs are usually *correlated*, not *independent* from each other. For example, in anti-cancer drug activity prediction tasks, chemical compounds which are active to one type of cancer are

more likely to be active to some other related cancers. It is much desirable that the correlations between different labels be exploited in the feature selection process.

3(a) illustrates the process of directly applying single-label graph feature selection methods by adopting the popular one-versus-all binary decomposition. The problems with this approach are as follows:

- multiple sets of discriminative subgraph features, one for each label or label combination, should to be mined before the classification, which could be too expensive when the number of labels is large;
- the correlations among multiple labels of the graphs are ignored in the feature selection process. In addition, the correlations among labels may result in similar feature sets for different labels. Redundancies in these sets of discriminative subgraph features cause unnecessary time and memory costs, since many of the features are mined multiple times.

In this chapter, we introduce a novel framework to the above problems by mining subgraph features using multiple labels of graphs. Our framework is illustrated in 3(b). Different from existing single-label feature selection methods for graph data, our approach, called gMLC, can utilize multiple labels of graphs to find an optimal set of subgraph features for graph classification. We first derive an evaluation criterion for subgraph features, named gHSIC, based upon a given graph dataset with multiple labels. Then in order to avoid exhaustive enumeration of all subgraph features, we propose a branch-and-bound algorithm to efficiently search for optimal subgraph features by pruning the subgraph search space using multiple labels of graphs.

Label correlations can also be considered in our proposed framework. In order to evaluate our proposed model, we perform comprehensive experiments on real-world multi-label graph classification tasks, which consist three real-world multi-label graph classification tasks, built on 18 conventional binary graph classification datasets. The experiments demonstrate that our feature selection approach can effectively boost multi-label graph classification performances. Moreover, we show that gMLC is more efficient by pruning the subgraph search space using multiple labels.

The rest of the chapter is organized as follows. We start by a brief review on related work of graph feature selection and multi-label classification in Section 2.2. Then introduce the preliminary concepts, give the problem analysis and present the gHSIC criterion in Section 2.3 and Section 2.4; In Section 2.5, we derive a branch and bound algorithm gMLC based upon gHSIC. In Section 2.6, we discuss how to incorporate label correlations into the gMLC framework. Then Section 2.7 reports the experiment results. In Section 2.8, we conclude the chapter.

2.2 Related Work

To the best of our knowledge, this chapter is the first work addressing the multi-label feature selection problem for graph classification. Our work is related to both multi-label classification techniques and subgraph feature based graph mining. We briefly discuss both of them.

Multi-label learning deals with the classification problem where each instance can belong to multiple different classes simultaneously. Conventional multi-label approaches are focused on instances in vector spaces. One well-know type of approaches is binary relevance (one-vs-all technique (20)), which transforms the multi-label problem into multiple binary classification

problems, one for each label. ML-KNN(22) is one of the binary relevance methods, which extends the lazy learning algorithm, kNN, to a multi-label version. It employs label prior probabilities gained from each example's k nearest neighbors and use *maximum a posteriori* (MAP) principle to determine label set. Elisseeff and Weston (19) presented a kernel method RANK-SVM for multi-label classification, by minimizing a loss function named *ranking loss*. Extension of other traditional learning techniques have also been studied, such as probabilistic generative models (17; 23), decision trees (24), maximal margin methods (25; 26) and ensemble methods(27), *etc.*

Extracting subgraph features from graph data have also been investigated by many researchers. The goal of such approaches is to extract informative subgraph features from a set of graphs. Typically some filtering criteria are used. Upon whether considering the label information, there are two types of approaches: unsupervised and supervised. A typical evaluation criterion is frequency, which aims at collecting frequently appearing subgraph features. Most of the frequent subgraph feature extraction approaches are unsupervised. For example, Yan and Han develop a depth-first search algorithm: gSpan (28). This algorithm builds a lexicographic order among graphs, and maps each graph to an unique minimum DFS code as its canonical label. Based on this lexicographic order, gSpan adopts the depth-first search strategy to mine frequent connected subgraphs efficiently. Many other frequent subgraph feature extraction approaches have been developed, *e.g.* AGM (29), FSG (30), MoFa (31), FFSM (32), and Gaston (33). Supervised subgraph feature extraction approaches have also been proposed in literature,

such as LEAP (13), CORK (14), which look for discriminative subgraph patterns for graph classifications, and gSSC (34) for semi-supervised classification.

Our approach is also relevant to graph feature selection approaches based on Hilbert-Schmidt independence criterion (35), but there are significant differences between them. Previous graph feature selection approaches assume each graph object only has one label and they focus on evaluating subgraph features effectively using HSIC criterion and perform feature selection using frequent subgraph mining methods (gSpan) as black-boxes. However, our approach assumes that each graph can have multiple labels, and focuses on extracting good subgraph features efficiently by pruning the subgraph search space using branch and bound method inside gSpan. So, our method searches the pruned gSpan tree. In fact, we only generated and searched a much smaller tree than gSpan as the size of the search tree dominates the execution time.

2.3 Problem Formulation

Before presenting the feature selection model for multi-label graph classification, we first introduce the notations that will be used throughout this chapter. Multi-label graph classification is the task of automatically classifying a graph object into a subset of predefined classes. Let $\mathcal{D} = \{G_1, \cdots, G_n\}$ denote the entire graph dataset, which consists of n graph objects, represented as *connected graphs*. The graphs in $\mathcal{D}$ are labeled by $\{\boldsymbol{y}_1, \cdots, \boldsymbol{y}_n\}$, where $\boldsymbol{y}_i \in \{0,1\}^Q$ denotes the multiple labels assigned to G_i. Here Q is the number of all possible labels within a label concept set $\mathcal{C}$.

Definition 1 (Connected Graph) *A graph is represented as $G = (\mathcal{V}, E, \mathcal{L}, l)$, where $\mathcal{V}$ is a set of vertices $\mathcal{V} = \{v_1, \cdots, v_{n_v}\}$, $E \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, $\mathcal{L}$ is the set of labels for the*

vertices and the edges. $l : \mathcal{V} \cup E \rightarrow \mathcal{L}$*,* l *is a function assigning labels to the vertices and the edges. A connected graph is a graph such that there is a path between any pair of vertices.*

Definition 2 (Multi-label Graph) *A multi-label graph is a graph assigned with multiple class labels* $(G, \boldsymbol{y})$*, in which* $\boldsymbol{y} = [y^1, \cdots, y^Q] \in \{0,1\}^Q$ *denotes the multiple labels assigned to the graph* G*.* $y^k = 1$ *iff graph* G *is assigned with the* k*-th class label,* 0 *otherwise.*

Definition 3 (Subgraph) *Let* $G' = (\mathcal{V}', E', \mathcal{L}', l')$ *and* $G = (\mathcal{V}, E, \mathcal{L}, l)$ *be connected graphs.* G' *is a subgraph of* G $(G' \subseteq G)$ *iff there exist an injective function* $\psi : \mathcal{V}' \rightarrow \mathcal{V}$ *s.t. (1)* $\forall v \in \mathcal{V}'$*,* $l'(v) = l(\psi(v))$*; (2)* $\forall (u,v) \in E'$*,* $(\psi(u), \psi(v)) \in E$ *and* $l'(u,v) = l(\psi(u), \psi(v))$*. If* G' *is a subgraph of* G*, then* G *is a supergraph of* G'*.*

In our current solution, we focus on the subgraph-based graph classification problem, which assumes that a graph object G_i is represented as a binary vector $\boldsymbol{x}_i = [x_i^1, \cdots, x_i^m]^\top$ associated with a set of subgraph patterns $\{g_1, \cdots, g_m\}$. Here $x_i^k \in \{0,1\}$ is the binary feature of G_i corresponding to the subgraph pattern g_k, and $x_i^k = 1$ iff g_k is a subgraph of G_i $(g_k \subseteq G_i)$.

The key issue of feature selection for multi-label graph classification is how to find the most informative subgraph patterns from a given multi-label graph dataset. So, in this chapter, the studied research problem can be described as follows: in order to train an effective multi-label graph classifier, how to efficiently find a set of optimal subgraph features using multiple labels of graphs?

Mining the optimal subgraph features for multi-label graphs is a non-trivial task due to the following problems:

1) How to properly evaluate the usefulness of a set of subgraph features based upon multiple labels of graphs?

2) How to determine the optimal subgraph features within a reasonable amount of time by avoiding the exhaustive enumeration using multiple labels of the graphs? The subgraph feature space of graph objects are usually too large, since the number of subgraphs grows exponentially with the size of graphs. It is infeasible to completely enumerate all the subgraph features for a given graph dataset.

3) How to incorporate the correlations among different labels in the feature selection process?

In the following sections, we will first introduce the optimization framework for selecting informative subgraph features from multi-label graphs, and propose an efficient subgraph mining strategy using branch-and-bound to avoid exhaustive enumeration. Then we propose solutions to incorporate label correlations into the feature selection process.

2.4 Optimization Framework

In this section, we address the problem 1) discussed in Section 2.3 by defining the subgraph feature selection for multi-label graph classification as an optimization problem. The goal is to find an optimal set of subgraph features based on the multiple labels of graphs. Formally, let us introduce the following notations:

- $\mathcal{S} = \{g_1, g_2, \cdots, g_m\}$: a given set of subgraph features, which we use to predict a set of multiple labels for each graph object. Usually there is only a subset of the subgraph features $\mathcal{T} \subseteq \mathcal{S}$ relevant to the multi-label graph classification task.

- $\mathcal{T}^*$: the optimal set of subgraph features $\mathcal{T}^* \subseteq \mathcal{S}$.
- $\mathcal{E}(\mathcal{T})$: an evaluation criterion to estimate the usefulness of subgraph feature subsets $\mathcal{T}$.
- X: the matrix consisting binary feature vectors using $\mathcal{S}$ to represent the graph dataset $\{G_1, G_2, \cdots, G_n\}$. $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n] = [\boldsymbol{f}_1, \boldsymbol{f}_2, \cdots, \boldsymbol{f}_m]^\top \in \{0,1\}^{m \times n}$, where $X = [X_{ij}]_{m \times n}$, $X_{ij} = 1$ iff $g_i \subseteq G_j$.

We adopt the following optimization framework to select an optimal subgraph feature set:

$$\begin{aligned} \mathcal{T}^* &= \arg\max_{\mathcal{T} \subseteq \mathcal{S}} \mathcal{E}(\mathcal{T}) \\ \text{s.t.} \quad & |\mathcal{T}| \leq t, \end{aligned} \tag{2.1}$$

where t denotes the maximum number of feature selected, $|\cdot|$ is the size of the feature set. Similar optimization framework to select an optimal subgraph feature set has also been defined in the context of single-label graph feature selection in (14; 35). In Equation 2.1 the objective function has two parts: the evaluation criterion $\mathcal{E}$ and the subgraph features of graphs $\mathcal{S}$.

For evaluation criterion, we assume that the optimal subgraph features should have the following property, *i.e. Dependence Maximization*: Optimal subgraph features should maximize the dependence between the subgraph features of graph objects and their multiple labels. This indicates that two graph objects with similar sets of multiple labels are likely to have similar subgraph features. Similar assumptions have also been used for multi-label dimensionality reduction in vector spaces (36).

Many criteria that can be used as dependence evaluation between subgraph features and multiple labels. In this chapter, we derive a subgraph evaluation criterion for multi-label graph classification based upon a dependence evaluation criterion named Hilbert-Schmidt Independence Criterion (HSIC) (37). We briefly introduce the Hilbert-Schmidt Independence Criterion as a dependence measure between two variables in kernel space. In our case, the target is to derive a dependence measure between the graph objects using a set of subgraph features and their multiple labels. Suppose we have two reproducing kernel Hilbert spaces (RKHS) of functions $\mathcal{G}$ and $\mathcal{F}$, with feature mapping $\phi(G_i) \in \mathcal{G}$ and $\psi(\boldsymbol{y}_i) \in \mathcal{F}$. The corresponding kernel functions are denoted as $\langle \phi(G_i), \phi(G_j) \rangle_{\mathcal{G}} = k(G_i, G_j)$ and $\langle \psi(\boldsymbol{y}_i), \psi(\boldsymbol{y}_j) \rangle_{\mathcal{F}} = k'(\boldsymbol{y}_i, \boldsymbol{y}_j)$. Let C be a covariance operator defined as

$$C = \mathrm{E}\left\{ [p(G_i) - \mathrm{E}(p(G_i))][p'(\boldsymbol{y}_i) - \mathrm{E}(p'(\boldsymbol{y}_i))] \right\}$$

for all $p \in \mathcal{G}$ and $p' \in \mathcal{F}$.

Then the HSIC is defined as the Hilbert-Schmidt norm of the operator C, *i.e.* $\|C\|^2_{HS}$. Given a sample of data, an empirical estimate of HSIC is HSIC $= \mathrm{tr}(\mathbf{K}\ \mathbf{H}\ \mathbf{L}\ \mathbf{H})$, where $\mathrm{tr}(\cdot)$ is the trace of matrix and $\mathbf{H} = [H_{ij}]_{n \times n}$, $H_{ij} = \delta_{ij} - 1/n$, δ_{ij} is the indicator function which takes 1 when $i = j$ and 0 otherwise. $\mathbf{K}$ and $\mathbf{L}$ are kernel matrices on the samples with respect to the kernel functions $k(\cdot, \cdot)$ and $l(\cdot, \cdot)$.

There are basically two reasons for using HSIC measure for feature selection:

- The HSIC can evaluate the dependence of two variables in kernel space, which is more general than measuring dependence in the original space. HSIC has been widely used for feature selection on single-label cases. It can also be extended to feature selection in multi-label cases. Moreover, correlations among different labels can naturally be considered in our framework by adopting advanced kernels into the HSIC. Thus it is more effective and flexible to measure the dependence in the kernel space.
- In addition to many good theoretical properties, HSIC has a very simple empirical estimator, tr(**KHLH**), which we can use to estimate the dependencies between input and output variables. The feature selection problem corresponds to selecting a subset of features such that the dependence between the input of the graph objects and the outputs (multiple labels) are maximized.

According to our *Dependence Maximization* assumption on the optimal subgraph features for multi-label graph classification, we can adopt the HSIC criterion to evaluate the dependence between the graph objects using a set of subgraph features and their multiple label outputs. Suppose we select a set of subgraph features $\mathcal{T}$, and each graph object G_i can be mapped into a feature space $\mathcal{G}$ by $\phi(G_i) = D_{\mathcal{T}}\boldsymbol{x}_i$ with the kernel function $k(G_i, G_j) = \langle\phi(G_i), \phi(G_j)\rangle = \langle D_{\mathcal{T}}\boldsymbol{x}_i, D_{\mathcal{T}}\boldsymbol{x}_j\rangle$. Here $D_{\mathcal{T}} = \mathrm{diag}(\boldsymbol{\delta}_{\mathcal{T}})$ is a diagonal matrix indicating which features are selected into feature set $\mathcal{T}$ from $\mathcal{S}$. And $\boldsymbol{\delta}_{\mathcal{T}} = [\delta^1_{\mathcal{T}}, \delta^2_{\mathcal{T}}, \cdots, \delta^m_{\mathcal{T}}]^\top \in \{0,1\}^m$ is an indicator vector, and $\delta^i_{\mathcal{T}} = 1$ iff $g_i \in \mathcal{T}$. Then the kernel matrix on the graph objects with subgraph features $\mathcal{T}$ is denoted as $\mathbf{K}_{\mathcal{T}}$. Suppose $\mathbf{L} = [L_{ij}]_{n\times n}$ is a kernel matrix based upon the multiple labels of each graph, and the kernel function is $l(\boldsymbol{y}_i, \boldsymbol{y}_j) = \langle\psi(\boldsymbol{y}_i), \psi(\boldsymbol{y}_j)\rangle$. In our current implementation,

$l(\boldsymbol{y}_i, \boldsymbol{y}_j) = \langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle$ is used as the default label kernel. Other kernels can also be directly used, which will be discussed in Section 2.6. Then we can evaluate the dependence between graph objects using feature set $\mathcal{T}$ and the multiple labels as follows:

$$\text{HSIC} = \text{tr}(\mathbf{K}_{\mathcal{T}}\mathbf{HLH})$$

The subgraph feature selection task corresponds to the selection of a subset of features in $\mathcal{S}$, such that the dependence between graph objects and their multiple labels are maximized.

In detail, we can rewrite the optimization problem in Equation 2.1 as follows:

$$\begin{aligned} \underset{\mathcal{T} \subseteq \mathcal{S}}{\arg\max} \quad & \text{tr}\ (\mathbf{K}_{\mathcal{T}}\mathbf{H}\ \mathbf{L}\ \mathbf{H}) \\ \text{s.t.} \quad & |\mathcal{T}| \leq t, \end{aligned} \tag{2.2}$$

The formula in Equation 2.2 can be rewritten as follows:

$$\begin{aligned} & \text{tr}\left(\mathbf{K}_{\mathcal{T}}\mathbf{HLH}\right) = \text{tr}\left(X^\top {D_{\mathcal{T}}}^\top D_{\mathcal{T}} X\mathbf{HLH}\right) = \text{tr}\left(D_{\mathcal{T}} X\mathbf{HLH}X^\top {D_{\mathcal{T}}}^\top\right) \\ = & \sum_{g_i \in \mathcal{T}} \left({\boldsymbol{f}_i}^\top \mathbf{HLH} \boldsymbol{f}_i\right) = \sum_{g_i \in \mathcal{T}} \left({\boldsymbol{f}_i}^\top \mathbf{M} \boldsymbol{f}_i\right) \end{aligned}$$

where $\mathbf{M} = \mathbf{HLH}$. By denoting function $h(g_i, \mathbf{M}) = {\boldsymbol{f}_i}^\top \mathbf{M} \boldsymbol{f}_i$, the optimization (Equation 2.2) can be written as

$$\begin{aligned} \max_{\mathcal{T}} \quad & \sum_{g_i \in \mathcal{T}} h(g_i, \mathbf{M}) \\ \text{s.t.} \quad & \mathcal{T} \subseteq \mathcal{S}, |\mathcal{T}| \leq t \end{aligned} \tag{2.3}$$

Definition 4 (gHSIC) *Suppose we have a multi-labeled graph dataset* $\mathcal{D} = \{(G_1, \boldsymbol{y}_1), \cdots, (G_n, \boldsymbol{y}_n)\}$. *Let* L *be a kernel matrix defined on the multiple label vectors, and* $\boldsymbol{M} = \boldsymbol{HLH}$. *We define a quality criterion* q *called gHSIC, for a subgraph feature* g *as*

$$q(g) = h(g, \boldsymbol{M}) = \boldsymbol{f}_g{}^\top \boldsymbol{M} \boldsymbol{f}_g$$

where $\boldsymbol{f}_g = [f_g^{(1)}, \cdots, f_g^{(n)}]^\top \in \{0,1\}^n$ *is the indicator vector for subgraph feature* g, $f_g^{(i)} = 1$ *iff* $g \subseteq G_i$ $(i = 1, 2, \cdots, n)$. *Since matrix* $\boldsymbol{L}$ *and* $\boldsymbol{M}$ *are positive semi-definite, for any subgraph pattern* g, *we have* $q(g) \geq 0$.

The optimal solution to the problem in Equation 2.2 can be found by using gHSIC to forward feature selection on a set of subgraphs $\mathcal{S}$. Suppose the gHSIC values for all subgraphs are denoted as $q(g_1) \geq q(g_2) \geq \cdots \geq q(g_m)$ in sorted order. Then the optimal solution to the optimization problem in Equation 2.3 is:

$$\mathcal{T}^* = \{g_i | i \leq t\}.$$

2.5 The Proposed Solution

Now we address the second problem discussed in Section 2.3, and propose an efficient method to find the optimal set of subgraph features from a given multi-label graph dataset.

Exhaustive enumeration: One of the most simple and straightforward solution for finding an optimal feature set is the exhaustive enumeration, *i.e.*, we first enumerate all subgraph patterns in a multi-label graph dataset, and then calculate the gHSIC values for all subgraph patterns.

However, in the context of graph classification, the number of subgraphs grows exponentially with the size of graphs, which makes the exhaustive enumeration approach usually impractical in real-world data.

Inspired by recent advances in graph classification approaches, *e.g.* (13; 34), which put their evaluation criteria into the subgraph pattern mining steps and develop constraints to prune search spaces, we take a similar approach by deriving a different constraint for multi-label cases. In order to avoid the exhaustive search, we proposed a branch-and-bound algorithm, named gMLC, which is summarized as follows: a) Adopt a canonical search space where all the subgraph patterns can be enumerated. b) Search through the space, and find the optimal subgraph features by gHSIC. c) Propose an upper bound of gHSIC and prune the search space.

2.5.1 Subgraph Enumeration

In order to enumerate all subgraphs from a graph dataset, we adopted an efficient algorithm, gSpan, proposed by Yan et al(28). We briefly review the general idea of gSpan approach: Instead of enumerating subgraphs and testing for isomorphism, they first build a lexicographic order over all the edges of a graph, and then map each graph to an unique minimum DFS code as its canonical label. The minimum DFS codes of two graphs are equivalent iff they are isomorphic. Details can be found in (28). Based on this lexicographic order, a depth-first search (DFS) strategy is used to efficiently search through all the subgraphs in a DFS code tree. By a depth-first search through the DFS code tree's nodes, we can enumerate all the subgraphs of a graph in their DFS code's order. And the nodes with non-minimum DFS codes can be directly pruned in the tree, which saves us from performing an explicit isomorphic test among the subgraphs.

2.5.2 Upper Bound of gHSIC

Now, we can efficiently enumerate all the subgraph patterns of a graph dataset in a canonical search space using gSpan's DFS Code Tree. Then, we derive an upper bound for the gHSIC value which can be used to prune the search space as follows:

Theorem 1 (Upper bound of gHSIC) *Given any two subgraphs $g, g' \in \mathcal{S}$, g' is a supergraph of g ($g' \supseteq g$). The gHSIC value of g' ($q(g')$) is bounded by $\hat{q}(g)$ (i.e., $q(g') \leq \hat{q}(g)$), where $\hat{q}(g)$ is defined as follows:*

$$\hat{q}(g) = \boldsymbol{f}_g{}^\top \hat{\boldsymbol{M}} \boldsymbol{f}_g \tag{2.4}$$

where the matrix $\hat{\boldsymbol{M}} = [\hat{M}_{ij}]_{n\times n}$ is defined as $\hat{M}_{ij} = \max(0, M_{ij})$. $\boldsymbol{f}_g = \{I(g \subseteq G_i)\}_{i=1}^n \in \{0,1\}^n$ is a vector indicating which graphs in a graph dataset $\{G_1, \cdots, G_n\}$ contain the subgraph g, $I(\cdot)$ is the indicator function. Suppose the gHSIC value of g is $q(g) = \boldsymbol{f}_g{}^\top \boldsymbol{M} \boldsymbol{f}_g$.

Proof 1

$$q\left(g'\right) = \boldsymbol{f}_{g'}{}^\top \boldsymbol{M} \boldsymbol{f}_{g'} = \sum_{i,j:G_i,G_j \in \mathcal{G}(g')} M_{ij}$$

where $\mathcal{G}(g') = \{G_i | g' \subseteq G_i, 1 \leq i \leq n\}$. *Since* g' *is the supergraph of* g ($g' \supseteq g$), *according to anti-monotonic property, we have* $\mathcal{G}(g') \subseteq \mathcal{G}(g)$. *Also* $\hat{M}_{ij} = \max(0, M_{ij})$, *we have* $\hat{M}_{ij} \geq M_{ij}$ *and* $\hat{M}_{ij} \geq 0$. *So,*

$$\begin{aligned} q\left(g'\right) &= \sum_{i,j:G_i,G_j \in \mathcal{G}(g')} M_{ij} \\ &\leq \sum_{i,j:G_i,G_j \in \mathcal{G}(g')} \hat{M}_{ij} \\ &\leq \sum_{i,j:G_i,G_j \in \mathcal{G}(g)} \hat{M}_{ij} = \hat{q}\left(g\right) \end{aligned}$$

Thus, for any $g' \supseteq g$, $q(g') \leq \hat{q}(g)$.

2.5.3 Subgraph Search Space Pruning

In this subsection, we make use of the the upper bound of gHSIC to efficiently prune the DFS Code Tree using a *branch-and-bound* method, which is similar to (34) but under different problem context: In depth-first search through the DFS Code Tree, we maintain the temporally suboptimal gHSIC value (denoted by θ) among all the gHSIC values calculated before. If $\hat{q}(g) < \theta$, the gHSIC value of any supergraph g' ($g' \supseteq g$) is no greater than θ. Now, we can safely prune the subtree from g in the search space. If $\hat{q}(g) \geq \theta$, we can not prune this space since there might exist a supergraph $g' \supseteq g$ ($q(g') \geq \theta$).

Figure 4 shows the algorithm gMLC. We first initialize the subgraphs $\mathcal{T}$ as an empty set. Then we prune the search space by running gSpan, while always maintaining the top-t best subgraphs according to q. In the course of mining, whenever we search to a subgraph g with $\hat{q}(g) \leq \min_{g_i \in \mathcal{T}} q(g_i)$, such that for any supergraph $g' \supseteq g$ ($q(g') \leq \hat{q}(g)$) according to the bound defined in Equation 2.4, we can prune the branches of the search tree originating from g . In

$\mathcal{T}$ = gMLC($\mathcal{D}$, *min_sup*, t)

Input:

$\mathcal{D}$: Multi-label graphs $\{(G_1, \boldsymbol{y}_1), \cdots, (G_n, \boldsymbol{y}_n)\}$
min_sup : Minimum support threshold
t : Maximum number of subgraph feature selected

Process:

$\mathcal{T} = \emptyset$, $\theta = 0$;
Recursively visit the DFS Code Tree in gSpan:
  g = currently visited subgraph in DFS Code Tree
  if $|\mathcal{T}| < t$, then
    $\mathcal{T} = \mathcal{T} \cup \{g\}$;
  else if $q(g) > \min_{g' \in \mathcal{T}} q(g')$, then
    $g_{min} = \mathrm{argmin}_{g' \in \mathcal{T}} q(g')$ and $\mathcal{T} = \mathcal{T}/g_{min}$;
    $\mathcal{T} = \mathcal{T} \cup \{g\}$ and $\theta = \min_{g' \in \mathcal{T}} q(g')$;
  if $\hat{q}(g) > \theta$ and $freq(g) \geq min\_sup$, then
    Depth-first search subtree rooted from node g;
return $\mathcal{T}$;

Output:

$\mathcal{T}$: Set of optimal subgraph features

Figure 4. The gMLC algorithm

the other hand, as long as the resulting subgraph g can still improve the gHSIC value of any subgraph $g_i \in \mathcal{T}$, it is accepted into $\mathcal{T}$ and the last best subgraph is dropped off from $\mathcal{T}$.

Note that in our experiments with the three datasets, the gHSIC criterion based on multiple labels provides such a bound that we can even omit the support threshold min_sup and still find a set of optimal subgraphs within a reasonable time cost.

2.6 Exploiting Label Correlations

Now we address the third problem discussed in Section 2.3, and explain how label correlations can be considered in gMLC framework by adopting more informative and advanced kernels.

In the previous sections, we used the simple kernel function, $l(\boldsymbol{y}_i, \boldsymbol{y}_j) = \langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle$, to generate the label kernel matrix $\mathbf{L}$. The linear kernel treats each label as being independent without considering the correlations among different labels. However in many real world applications, the multiple labels of the graphs are usually correlated. For example, in anti-cancer drug activity prediction tasks, chemical compounds which are active to one type of cancer are more likely to be active to some other related cancers. Subgraph patterns that corresponds to such label co-occurrences can be very useful for multi-label graph classification. In order to put label correlations into consideration during feature mining process, we need to adopt more informative kernels for $\mathbf{L}$ than linear kernel.

One simple solution is that the label correlations can be exploited by adopting more advanced kernels like polynomial or RBF kernels in the label kernel calculation. *i.e.*, the label vec-

tor $\boldsymbol{y}$ is mapped to a new feature space using $\psi(\boldsymbol{y})$ with kernel function $l(\boldsymbol{y}_i, \boldsymbol{y}_j) = \langle \psi(\boldsymbol{y}_i), \psi(\boldsymbol{y}_j) \rangle$, and the correlations among different labels are explicitly considered in the new feature space.

For example, suppose we use a polynomial kernel with degree 2, $l(\boldsymbol{y}_i, \boldsymbol{y}_j) = \langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle^2$, as the label kernel function. Given any two label vectors $\boldsymbol{\alpha} = [\alpha_1, \alpha_2] \in \{0, 1\}^2$ and $\boldsymbol{\beta} = [\beta_1, \beta_2] \in \{0, 1\}^2$, we have

$$\begin{aligned} l(\boldsymbol{\alpha}, \boldsymbol{\beta}) &= \langle \boldsymbol{\alpha}, \boldsymbol{\beta} \rangle^2 \\ &= (\alpha_1 \beta_1 + \alpha_2 \beta_2)^2 \\ &= \left\langle \left[{\alpha_1}^2, {\alpha_2}^2, \sqrt{2}\alpha_1 \alpha_2 \right] \ , \ \left[{\beta_1}^2, {\beta_2}^2, \sqrt{2}\beta_1 \beta_2 \right] \right\rangle \\ &= \langle \psi(\boldsymbol{\alpha}), \ \psi(\boldsymbol{\beta}) \rangle \end{aligned}$$

Here, $\psi(\boldsymbol{\alpha}) = \left[{\alpha_1}^2, {\alpha_2}^2, \sqrt{2}\alpha_1 \alpha_2 \right]$, and the component $(\sqrt{2}\alpha_1 \alpha_2)$ considers the correlations between label l_1 and l_2 explicitly. Intuitively, by adopting polynomial kernels with degree 2, the *second-order* correlations among different labels can be exploited in our gMLC framework. Higher orders of correlations among labels can also be exploited by adopting polynomial kernels with higher degrees or even RBF kernels to construct the label kernel $\mathbf{L}$.

After using these kernel functions, the new label kernel matrix $\mathbf{L}$ can be directly plugged in the subgraph evaluation criterion, $q(g) = {\boldsymbol{f}_g}^\top \mathbf{HLH} \boldsymbol{f}_g$. Subgraph patterns that best corresponds to the co-occurrence of different labels will get high values, thus being selected into the optimal feature set for multi-label graph classification.

TABLE I. Summary of experimental tasks studied. "AvgL" denotes the average number of labels assigned to each graph.

Prediction Task	Dataset	# Graphs	# Labels	AvgL
Anti-cancer	NCI1	812	10	4.36
Toxicology	PTC	253	4	1.60
Kinase Inhibition	NCI2	5,660	4	1.04

TABLE II. Details of the anti-cancer activity prediction task (NCI1 dataset). Each label represents the assay result for one type of cancer. "Pos (%)" denotes the average percentage of positive instances for each cancer assay.

Assay ID	Class Name	Pos (%)	Cancer Type
1	NCI-H23	35.6	Non-Small Cell Lung
33	UACC-257	47.7	Melanoma
41	PC-3	38.5	Prostate
47	SF-295	34.1	Central Nerve System
81	SW-620	17.5	Colon
83	MCF-7	59.2	Breast
109	OVCAR-8	42.2	Ovarian
123	MOLT-4	73.5	Leukemia
145	SN12C	54.8	Renal
330	P388	33.4	Leukemia

TABLE III. Details of toxicology prediction task (PTC dataset), where each of the multiple labels represents the toxicology test result on one type of animal. "Pos (%)" denotes the average percentage of positive instances for each cancer assay.

Class Name	Pos (%)	Animal Model
MR	41.9	Male Rats
FR	36.0	Female Rats
MM	38.7	Male Mice
FM	43.1	Female Mice

TABLE IV. Details of kinase inhibition prediction task (NCI2 dataset), where each of the multiple labels represents the inhibition of one type of kinase. "Pos (%)" denotes the average percentage of positive instances for each cancer assay.

Assay ID	Pos (%)	Kinase Type
1416	6.11	PERK
1446	40.5	JAK2
1481	15.9	ATPase
1531	41.4	MEK

2.7 Experiments

2.7.1 Experimental Setup

2.7.1.1 Data Collections

In order to evaluate the multi-label graph classification performances, we tested our algorithm on three real-world multi-label graph classification tasks as follows: (Summarized in Table I.)

1) Anti-cancer activity prediction (NCI1): The first task is to classify chemical compounds' anti-cancer activities on multiple types of cancer. We build up a multi-label graph dataset using a benchmark dataset, NCI[1] (13), which consists of records of chemical compounds' anti-cancer activities against a set of 10 types of cancer (*e.g.* Leukemia, Prostate, Breast), and each chemical compound is represented as a graph. After removing compounds with incomplete records for 10 types of cancer, we thus have a multi-label graph classification dataset with 812 graphs assigned with 10 candidate labels. Table II provides a brief description of the 10 types of cancer in NCI1 dataset.

2) Toxicology prediction of chemical compounds (PTC): The second task is to classify chemical compounds' carcinogenicity on multiple animal models. We build up our second multi-label graph dataset using a benchmark dataset, PTC[2] (38), which consists carcinogenicity records of 417 chemical compounds on 4 animal models: MM (Male Mouse),

---

[1]http://pubchem.ncbi.nlm.nih.gov

[2]http://www.predictive-toxicology.org/ptc/

FM (Female Mouse), MR (Male Rat) and FR (Female Rat). Each chemical compound is assigned with carcinogenicity labels for the 4 animal models. On each animal model the carcinogenicity label is one of {CE, SE, P, E, EE, IS, NE, N}. We assume {CE, SE, P} as 'positive' labels, {NE, N} as 'negative' and { E, EE IS} labels are removed, which is the same setting as (39; 21). Each chemical compound is represented as a graph with an average of 25.7 vertices. After removing compounds with incomplete records for the 4 animal models, we thus have a multi-label graph classification dataset with 253 graphs assigned with four candidate labels (MR, FR, MM, FM). Table III provides a brief description of the 4 animal models in PTC dataset.

3) Kinase inhibition prediction of chemical compounds (NCI2): The third task is to classify the ability of chemical compounds to inhibit multiple kinases' activity, which is a important problem in finding effective inhibitors for kinase associated diseases (*e.g.* infectious diseases, cancers). We build up our third multi-label graph dataset also from NCI database, which consists kinase inhibition records of 5,660 chemical compounds against a set of 4 types of kinases (*i.e.* ATPase, PERK, MEK, JAK2). After removing compounds with incomplete records for the 4 types of kinases, we thus have a multi-label graph classification dataset with 5,660 graphs assigned with 4 candidate labels. Table IV provides a brief description of the 4 types of kinases in NCI2 dataset.

2.7.1.2 Evaluation Metrics

Multi-label classification requires different evaluation metrics than conventional single-label classification problems. Here we adopt some metrics used in (18; 19; 22) to evaluate the

multi-label graph classification performance. Assume we have a multi-label graph dataset $\mathcal{D} = \{(G_1, \boldsymbol{y}_1), \cdots, (G_n, \boldsymbol{y}_n)\}$, where graph G_i is labeled as $\boldsymbol{y}_i \in \{0,1\}^Q$. Let $f(G_i, k)$ denote the classifier's real-value outputs for G_i on the k-th label (l_k), and $h(G_i) \in \{0,1\}^Q$ denotes the classifier's binary output label vector. According to $f(G_i, k)$ we can define a ranking function $rank_f(G_i, k) \in \{1, 2, \cdots, Q\}$, and $rank_f(G_i, k') < rank_f(G_i, k)$ iff $f(G_i, k') < f(G_i, k)$. We have the following evaluation criteria:

- Ranking Loss (19): evaluates the performance of classifier's real-value outputs $f(G_i, k)$. It is calculated as the average fraction of incorrectly ordered label pairs:

$$RankLoss = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\mathbf{1}^\top \boldsymbol{y}_i \mathbf{1}^\top \overline{\boldsymbol{y}}_i} Loss_f(G_i, \boldsymbol{y}_i)$$

  Where the $\overline{\boldsymbol{y}}_i$ denotes the complementary of $\boldsymbol{y}_i$ in $\{0,1\}^Q$.

$$Loss_f(G_i, \boldsymbol{y}_i) = \sum_{k:y_i^k=1}\sum_{k':y_i^{k'}=0} [\![f(G_i, k) \leq f(G_i, k')]\!]$$

  For any predicate π, $[\![\pi]\!]$ equals 1 if π holds and 0 otherwise. $RankLoss \in [0,1]$. The smaller the value, the better the performance.

- Average Precision (22): evaluates the average fraction of labels ranked above a particular label y s.t. y is in the ground-truth label set. This criterion is originally used in infor-

mation retrieval (IR) systems to evaluate the document ranking performance for query retrieval:

$$AvgPrec = \frac{1}{n}\sum\nolimits_{i=1}^{n}\frac{1}{\mathbf{1}^{\top}\boldsymbol{y}_i}\sum_{k:y_i^k=1}\frac{\mathrm{Prec}_f(G_i,k)}{rank_f(G_i,k)}$$

which measure the number of assigned class labels that are ranked before k-th class. Here

$$\mathrm{Prec}_f(G_i,k) = \sum_{k':y_i^{k'}=1}[\![rank_f(G_i,k') \leq rank_f(G_i,k)]\!]$$

And $AvgPrec \in [0,1]$, the larger the value, the better the performance.

- *One error*: evaluates how many times the top-ranked label is not in the set of ground-truth labels of the instance.

$$OneError = \frac{1}{n}\sum\nolimits_{i=1}^{n}[\![y_i^{k_i}=0]\!]$$

where $k_i = \mathrm{argmax}_{k\in[1,Q]} f(G_i,k)$. $OneError \in [0,1]$, the smaller the value, the better the performance.

- *Coverage*: evaluates the performance by considering how far, on average, we need to go down the ranked label list to cover all the ground-truth labels of the instance.

$$Coverage = \frac{1}{n}\sum\nolimits_{i=1}^{n}\max_{k:y_i^k=1} rank_f(G_i,k) - 1$$

where $Coverage \in [0, Q-1]$. The smaller the coverage, the better the performance.

- *Hamming loss*: evaluates how many times an instance-label pair is misclassified. For single-label problems, it equals the classification error.

$$HammingLoss = \frac{1}{n}\sum_{i=1}^{n}\theta(h(G_i), \boldsymbol{y}_i)$$

where

$$\theta(h(G_i), \boldsymbol{y}_i) = \frac{1}{Q}\sum_{k=1}^{Q}[\![y_i^k \neq h(G_i)^k)]\!]$$

and $HammingLoss \in [0, 1]$, the smaller the value, the better the performance.

In our experiment, we will show the value of $1 - AvePrec$ instead of *Average Precision*. Thus under all these evaluation criteria, smaller values are all indicating better performances. Note that all the criteria evaluate the performance of multi-label classification systems from different aspects. Usually few algorithms could outperform another algorithm on all those criteria. All experiments are conducted on machines with 4 GB RAM and Intel Xeon^{TM}Quad-Core CPUs of 2.40 GHz.

2.7.1.3 Comparing Methods

In order to demonstrate the effectiveness of our multi-label graph feature selection approach, we test with following methods:

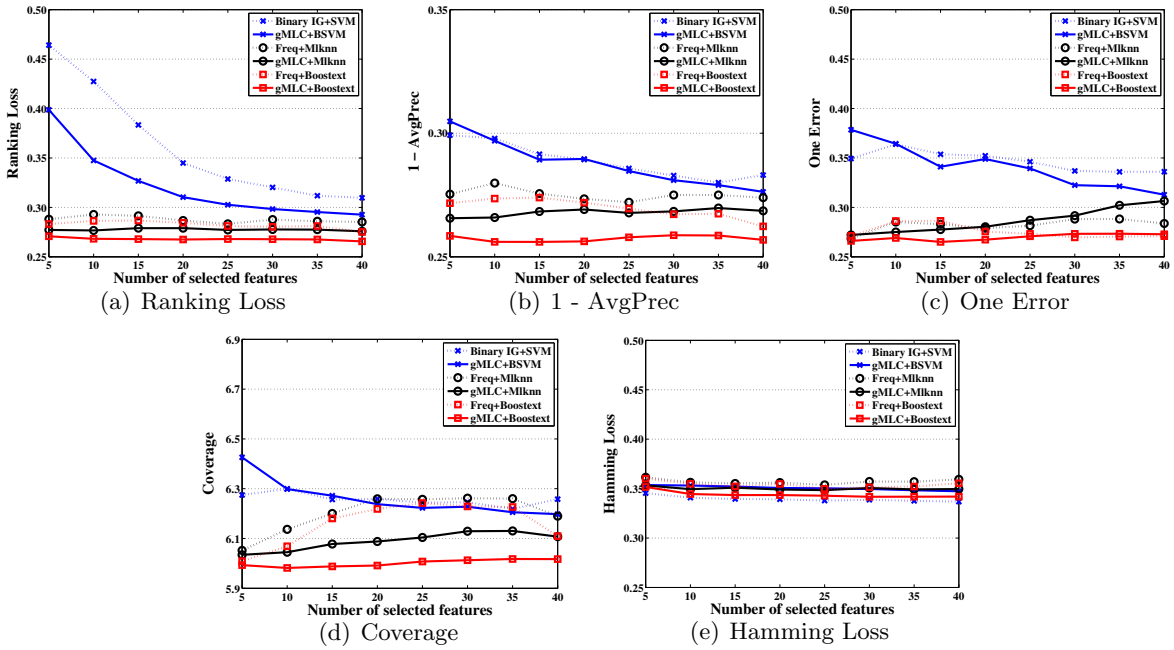
- Binary decomposition + single-label feature selection + binary classifications (Binary IG+ SVM): We first compare with a baseline using a binary decomposition method similar to (20): The multi-label graph dataset is first divided into multiple single-label graph datasets by one-vs-all binary decomposition. For each binary classification task, we use

the Information Gain (IG), an entropy based measure, to select a subset of discriminative features from frequent subgraphs. Then SVMs are used as the binary classification models to classify the graphs into multiple binary classes respectively. We use SVM-light software package[1] to train the SVMs, where the parameters are set as default settings.

- Multi-label feature selection (gMLC) + binary classifications (SVM): gMLC is used to find a set of optimal subgraph features. Then the one-vs-all deduction with one SVM trained for each class is used as the multi-label classifier.
- Top-k frequent subgraph features (Freq) + multi-label classification (BoosTexter): We also compare with another baseline: multi-label classification using the top-k frequent subgraphs as features, *i.e.*, we use the top-k frequent subgraph features in the graph dataset without the gHSIC selections on the subgraph features. Then BoosTexter(18) is used as the multi-label classifier. The number of boosting rounds for BoosTexter is set as 500, which does not significantly affect the classification performance.
- Multi-label feature selection (gMLC) + multi-label classification (BoosTexter): gMLC is used to find a set of optimal subgraph features. Then BoosTexter is used as the multi-label classifier.
- Top-k frequent features (Freq) + multi-label classification (Ml-knn): multi-label classification using the top-k frequent subgraphs as features. Ml-knn (22) is used as the multi-label classifier. The number of neighbors is set as the default value 10.

---

[1] http://svmlight.joachims.org/



(a) Ranking Loss (b) 1 - AvgPrec (c) One Error

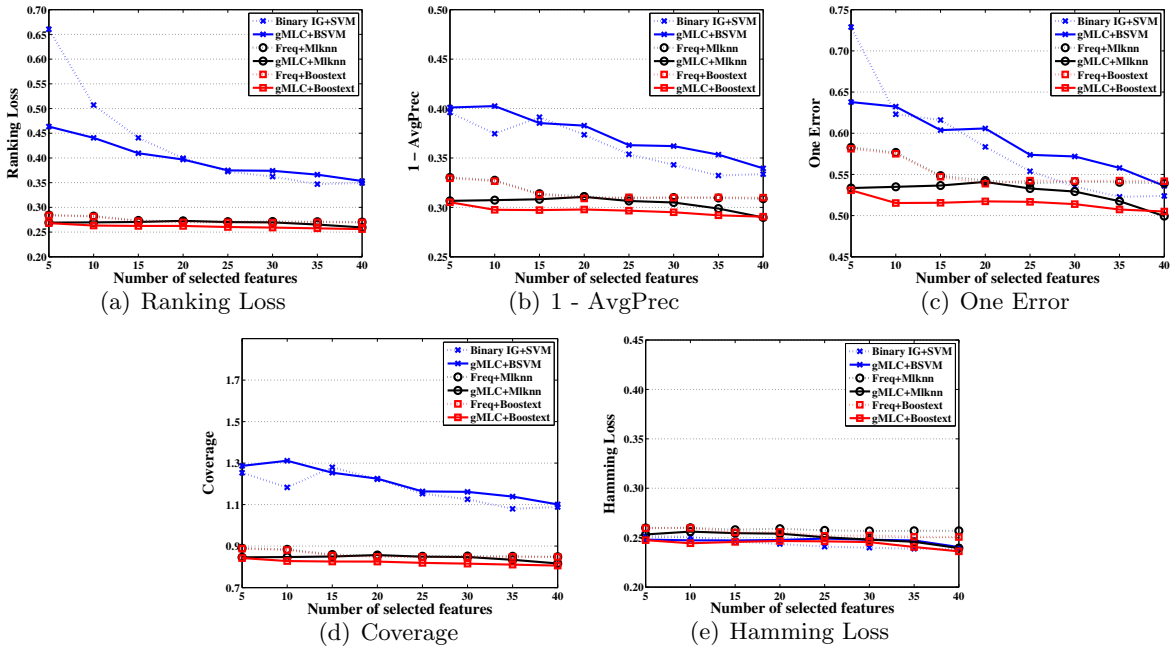
(d) Coverage (e) Hamming Loss

Figure 5. Multi-label graph classification performances on Anti-cancer Activity Prediction (NCI1 dataset)

- Multi-label feature selection (gMLC) + multi-label classification (ML-KNN): We first use gMLC to find a set of optimal subgraph features. Then ML-KNN is used as the multi-label classifier.

2.7.2 Performances on Multi-label Graph Classification

In our experiment, we use 10-round 10-fold cross validation to evaluate the multi-label graph classification performance. Each graph dataset is evenly partitioned into 10 parts. Only one part is used as testing graphs and the other nine are used as training graphs for frequent subgraph mining, feature selection and multi-label classification. We repeat the 10-fold cross



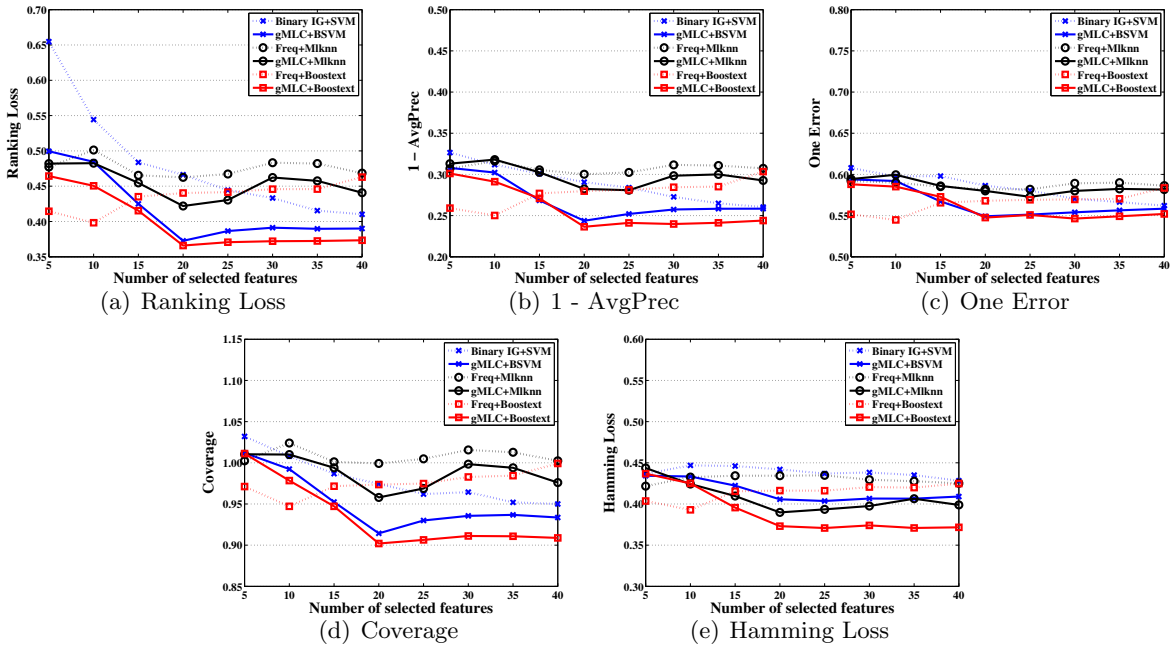
(a) Ranking Loss (b) 1 - AvgPrec (c) One Error

(d) Coverage (e) Hamming Loss

Figure 6. Multi-label graph classification performances on Kinase Inhibition Prediction (NCI2 dataset)

validation 10 times and we report the average results for the 10 rounds. The result of the feature selection methods for multi-label graph classification on NCI1, NCI2 and PTC datasets are displayed in Figure 5, Figure 6 and Figure 7. We show the number of selected subgraphs t among frequent subgraphs using $min\_sup = 10\%$, together with evaluation metrics mentioned before.

Now, we first study the effectiveness of selecting subgraph features by comparing two approaches: gMLC+SVM, Binary IG+ SVM, where the binary SVMs are used as base learners. It is worth noticing that, this comparison is only used for reference, since different number

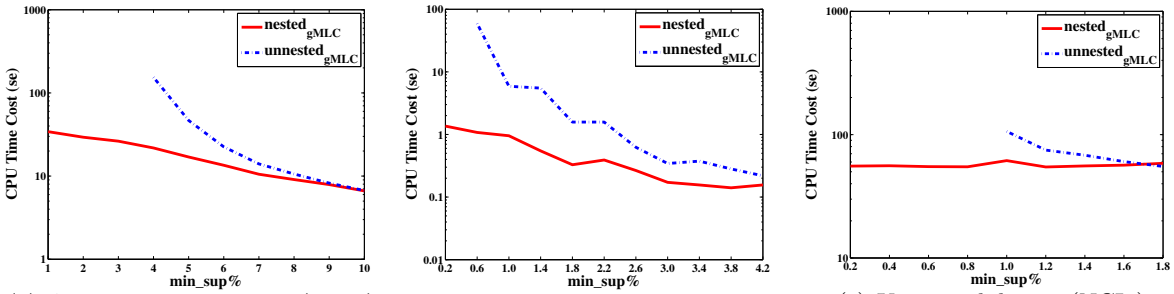


(a) Ranking Loss (b) 1 - AvgPrec (c) One Error

(d) Coverage (e) Hamming Loss

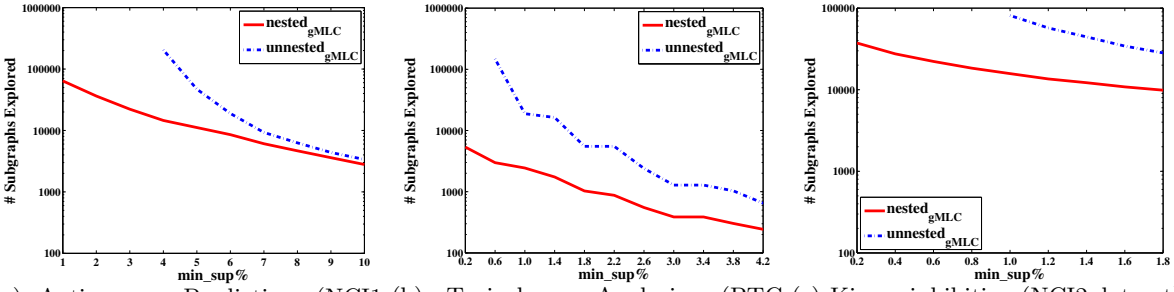
Figure 7. Multi-label graph classification performances on Toxicology Prediction Task (PTC dataset)

of features are used in the two methods. Our gMLC is designed for conventional multi-label classification methods, thus in the baseline gMLC+SVM, we select one set of subgraph features which is used on multiple SVMs separately. However, Binary IG+ SVM selects a different set of subgraph features for each label concept and these feature sets are used on multiple SVMs separately. Hence, Binary IG+ SVM method has an advantage over our method by using different feature sets for different SVMs, while gMLC uses the same set of feature for all the SVMs. Figure 5, Figure 6 and Figure 7 indicate that gMLC+SVM can achieve comparable or even better performances than Binary IG+ SVM in most cases. This is because the multiple labels



(a) Anti-cancer Prediction (NCI1) (b) Toxicology Analysis (PTC) (c) Kinase inhibition (NCI2)

Figure 8. Average CPU time for nested gMLC versus un-nested gMLC with varying min_sup.



(a) Anti-cancer Prediction (NCI1 dataset) (b) Toxicology Analysis (PTC dataset) (c) Kinase inhibition (NCI2 dataset)

Figure 9. Average number subgraph patterns explored during mining for nested gMLC versus un-nested gMLC with varying min_sup.

of the graphs usually have certain correlations, and the useful subgraph features on one label concept are also likely to be useful on some other label concepts. Thus our gMLC method can achieve better performances over Binary IG+ SVM even though we use a same set of feature for all binary SVMs. Utilizing the potential relations among multiple label concepts to select subgraph features are crucial to the success of our method in this case.

We further study the effectiveness of subgraph features using the general purposed multi-label classification methods, *i.e.* BoosTexter and Ml-knn, as the base classifiers. It is also worth noticing that, to the best of our knowledge, gMLC is the first multi-label feature selection method for graph data. Thus we cannot find any other baseline which select one set of features for multiple label concepts in order to make a fair comparison. So our only choices are comparing the following methods: gMLC+BoosTexter v.s. Freq+BoosTexter and gMLC+Ml-knn v.s. Freq+Ml-knn. We observe that on most tasks the performances of gMLC+BoosTexter are better than Freq+BoosTexter, *i.e.* multi-label classification approaches without gHSIC subgraph feature selection. Similar results can also be found with the cases when Ml-knn is used as the base classifier. These results support our intuition that the gHSIC evaluation criterion in gMLC can find better subgraph patterns for multi-label graph classification than unsupervised top-k frequent subgraph approaches. The exception is only the case on PTC dataset when the number of features selected is small (less than 15). Nonetheless, the Freq+BoosTexter can never reach the best performance achievable by gMLC with a larger number of features. This is because the top 15 frequent features happen to be good classification features. However, the Freq cannot find other good features that are not that frequent.

Now, we first study the effectiveness of selecting subgraph features by comparing two approaches: gMLC+SVM, Binary IG+ SVM, where the binary SVMs are used as base learners. It is worth noticing that, our gMLC is specially designed for conventional multi-label classification methods which require one set of features for all labels concepts. Thus gMLC only selects

one set of subgraph features and uses it on multiple SVMs separately. However, Binary IG+ SVM selects a different set of subgraph features for each label concept and these feature sets are used on multiple SVMs separately. Hence, Binary IG+ SVM method has an advantage over our method by using different feature sets for different SVMs, while gMLC uses the same set of feature for all the SVMs. Figure 5, Figure 6 and Figure 7 indicate that gMLC+SVM can achieve compariable or even better performances than Binary IG+ SVM in most cases. This is because the multiple labels of the graphs usually have certain correlations, and the useful subgraph features on one label concept are also likely to be useful on some other label concepts. Thus our gMLC method can achieve better performances over Binary IG+ SVM even though we use a same set of feature for all binary SVMs. Utilizing the potential relations among multiple label concepts to select subgraph features are crucial to the success of our method in this case.

We further observe that in all tasks and evaluation criteria, our multi-label feature selection algorithm with multi-label classification (gMLC+BoosTexter) outperforms the binary decomposition approach using single-label feature selections (Binary IG+ SVM). gMLC+BoosTexter can achieve good performances with only a small number of features. We note that the big improvement can both be counted on the good performance of gMLC feature selection and the state-of-the-art multi-label classification method, BoosTexter. However, this result can just be used for a reference to the relative performances of the two types of multi-label graph classification methods, binary decomposition based and gMLC based. These results support

the importance of the proposed multi-label feature selection method in the multi-label graph classification problems.

Additionally, by comparing over different evaluation criteria, we can find that gMLC shows more improvements over other baselines on criteria, *e.g.* Ranking Loss, which are most related to multi-label performances, than Hamming Loss. For Hamming Loss, gMLC gets better performances over other baselines on PTC dataset, but comparible performances on NCI1 and NCI2 dataset. This can be explained that Hamming Loss evaluates the classification performance in a binary way, simply averaging the binary classification error on each label without considering the ranking of all labels which is more important for multi-label classification evaluation.

2.7.3 Effectiveness of Subgraph Search Space Pruning

In our second experiment, we evaluated the effectiveness of the upper-bound for gHSIC proposed in Section 2.5.2. So, in this section we compare the runtime performance of two versions of implementation for gMLC: "nested gMLC" versus "un-nested gMLC". The "nested gMLC" denotes the proposed method using the upper-bound proposed in Section 2.5.2 to prune the search space of subgraph enumerations; the "un-nested gMLC" denotes the method without the gHSIC's upper-bound pruning, which first uses gSpan to find a set of frequent subgraphs, and then selects the optimal set of subgraphs via gHSIC. We run both approaches on the three tasks and record the average CPU time used on feature mining and selection. The result is shown in Figure 8.

In the NCI1, NCI2 and PTC dataset, we observe that as we decrease the min_sup in the frequent subgraph mining, the un-nested gMLC would need to explore larger subgraph search

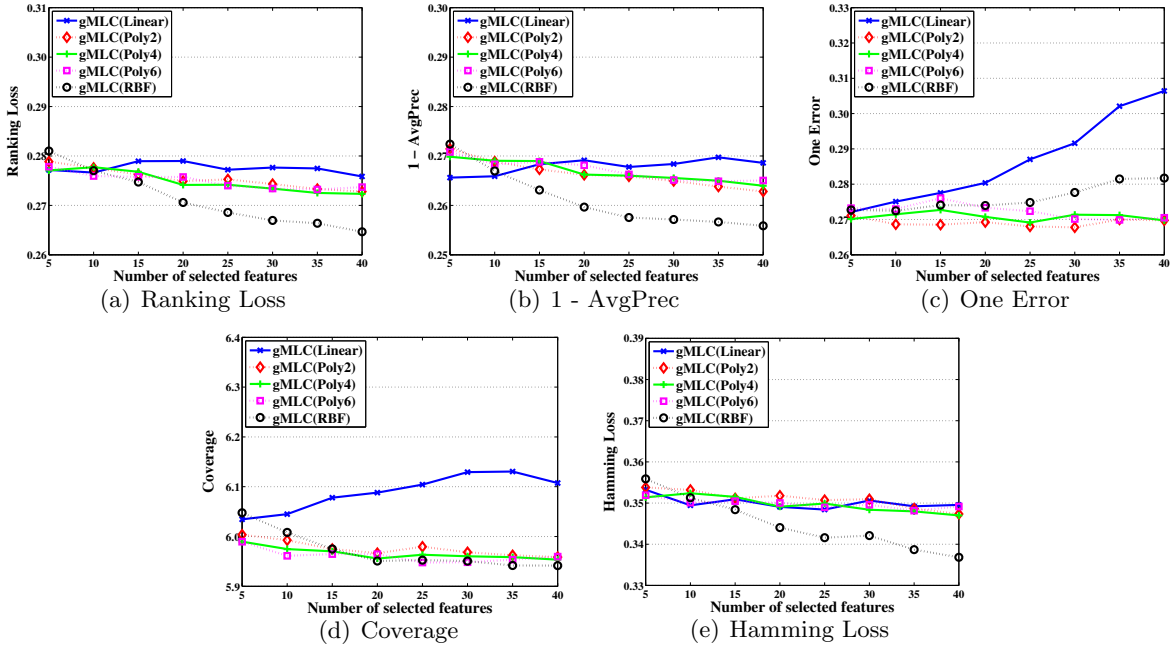
spaces, and this size increases exponentially with the decrease of *min_sup*. In the NCI1 dataset, when the *min_sup* get too low ($min\_sup < 4\%$), the subgraph feature enumeration step in un-nested gMLC can run out of the computer memory. However, the nested gMLC's running time does not increase as much, because the gHSIC can help pruning the subgraph search space using the multi-label information of the graphs. As we can see, the *min_sup* can go to very low value in all datasets for the "nested gMLC".

Figure 9 shows the number of subgraph feature explored in the process of subgraph pattern enumeration in the three tasks. In all tasks, we observe that the number of searched subgraph patterns in nested gMLC is much smaller than that of un-nested gMLC (the gSpan step). In our experiments, we further noticed that on most datasets, nested gMLC provides such a strong bound that we may even allow nested gMLC to omit the minimum support threshold *min_sup* and still receive an optimal set of subgraph features within a reasonable time.

2.7.4 Effectiveness of Embedding Label Correlations

In our third experiment, we evaluated the effectiveness of the label kernels after incorporating the label correlations in Section 2.6. In order to consider label correlations of first-order, second-order and higher-orders *etc.*, we use the following kernel functions to produce the label kernel matrix $\mathbf{L}$:

- gMLC(Linear) denotes our gMLC method with linear kernels for $\mathbf{L}$, which does not consider the label correlation. The kernel function is $l(\boldsymbol{y}_i, \boldsymbol{y}_j) = \langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle$.
- gMLC(Poly) denotes the gMLC method with polynomial kernels with different degrees, which can consider label correlations of second-orders or even higher-orders. The kernel



(a) Ranking Loss (b) 1 - AvgPrec (c) One Error

(d) Coverage (e) Hamming Loss

Figure 10. Performances of gMLC with/without considering label correlations on anti-cancer activity prediction task (NCI1 dataset)

function is $l(\boldsymbol{y}_i, \boldsymbol{y}_j) = (\gamma \langle \boldsymbol{y}_i, \boldsymbol{y}_j \rangle + \eta)^d$. The γ is set as the default value $\gamma = \frac{1}{\#\text{features}}$, and $\eta = 0$. d denotes the degree of polynomial kernels. For example, gMLC(Poly2) corresponds to the polynomial kernel with degree two ($d = 2$).

- gMLC(RBF) denotes our gMLC method with RBF kernels for $\mathbf{L}$, which can consider label correlations of any orders. The kernel function is $l(\boldsymbol{y}_i, \boldsymbol{y}_j) = \exp(-\gamma |\boldsymbol{y}_i - \boldsymbol{y}_j|^2)$. The γ is set as the default value $\gamma = \frac{1}{\#\text{features}}$.

In all methods, ML-KNN is used as the base classifier, with default parameter settings ($k = 10$). The result of NCI1 dataset is illustrated in Figure 10. From the results, we can see that gMLC

with polynomial kernel and RBF kernels can get better performances than gMLC with linear kernels, by considering label correlations in the label kernel matrix **L**. Here we only use simple strategies to consider label relationship in our gMLC model, and greater improvements are likely to be obtain by defining more advanced kernels for label matrix **L**.

2.8 Conclusion

In this chapter, we study the problem multi-label feature selection for graph classification. It is significantly more challenging than the conventional single-label feature selection in graph data because of the multiple labels assigned to each graph. To address this challenge, we propose an evaluation criterion gHSIC to evaluate the dependence of subgraph features with the multiple labels of graphs, and derived an upper-bound for gHSIC to prune the subgraph search space. Then we propose a branch-and-bound algorithm to efficiently find a compact set of subgraph feature which is useful for the classification of graphs with multiple labels. Empirical studies on real-world tasks show that our feature selection method for multi-label graph classification, gMLC, can effectively boost multi-label graph classification performances and is more efficient by pruning the subgraph search space using multiple labels. Additionally, the correlations among different labels can be exploited effectively by adopting more informative and advanced kernels for label kernel matrix.

In our current implementation, we only use simple strategies to construct label kernel matrix. Actually various other types of label kernels can also be used to exploit the label correlations among multiple labels more effectively. We will leave related discussions to potential future works.

CHAPTER 3

SEMI-SUPERVISED GRAPH CLASSIFICATION

3.1 Introduction

A major difficulty in graph classification lies in the complex structure of graphs and lack of vector representations. Selecting a proper set of features for graph data is an essential and important procedure for graph classification. The general problem of feature selection is well studied in the literature. Semi-supervised feature selection problem for graph data, however, has not been studied in this context so far. Conventional feature selection approaches on graph data assume, explicitly or implicitly, that there exists a large amount of labeled training data. However, in many real world applications, the labels of graph data are very expensive or difficult to obtain. Creating a large training dataset can be too expensive, time-consuming or even infeasible. For example, in molecular medicine, it requires time, efforts and excessive resources to test drugs' anti-cancer efficacies by pre-clinical studies and clinical trials, while there are often copious amounts of unlabeled drugs or molecules available from various sources.

Thus it is much desired that the large amounts of unlabeled graphs can be effectively utilized to select better features for graphs, and improve the graph classification performances. For example, in Figure 11, we show a dataset with two labeled graphs and four unlabeled graphs. Based only on the two labeled graphs, subgraph feature "a-b" and "a-c" are both discriminative features. Clearly, when we consider the distribution of the four unlabeled graphs, "a-b" is more

likely to be useful than "a-c". This is because the unlabeled graphs are not separable based on the subgraph feature "a-c".

Despite its value and significance, the semi-supervised feature selection for graph classification is a much more challenging task due to the specific characteristics of the task. The reasons are listed as follows.

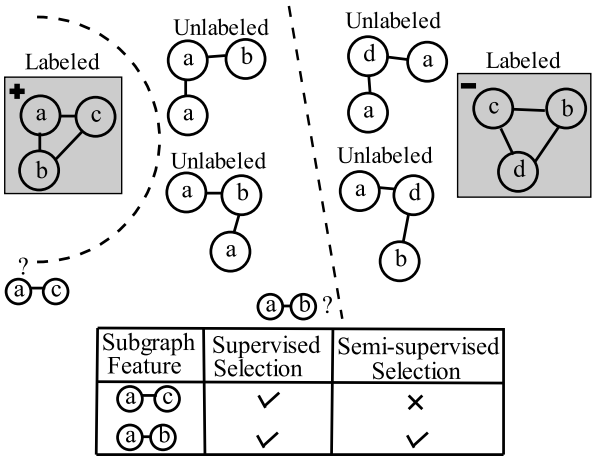


Figure 11. An example of semi-supervised feature selection on graph data. The subgraph feature "a-b" is more useful than "a-c" based on both labeled and unlabeled graphs.

1. *Lack of labels.* Conventional feature selection in graph classification approaches focuses on supervised settings (21; 13; 14). The mining strategy of discriminative subgraph patterns strictly follows the assumption that there exists a large amount of labeled graphs. How-

ever, many real-world graph classifications usually suffer from a lack of training graphs. It is usually laborious, or even infeasible to create a large training set of graph instances.

2. *Lack of features.* Another fundamental problem in semi-supervised feature selection on graph data lies in the complex structures and lack of feature representations of graphs. Conventional feature selection approaches in vector spaces, which assume a candidate feature set is available, cannot be directly applied to graph data, because it is usually infeasible to generate all the subgraph features of a graph dataset before feature selection. The number of subgraphs is usually too large to be fully generated, since it grows exponentially with the graph size. Furthermore checking subgraph isomorphism is NP-complete.

   In order to efficiently find discriminative subgraph features, conventional supervised subgraph feature mining approaches rely on the label information from a large training set to prune the subgraph search space and select useful features (13). However, when the number of labeled graphs is not large enough, the usefulness of the mined subgraph features can be weak, and the pruning of the subgraph mining process can be ineffective.

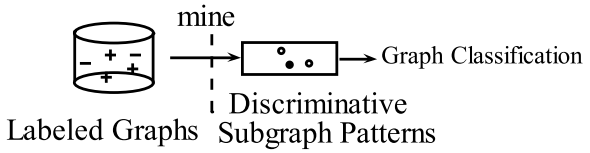
12(a) illustrates the feature selection process in conventional graph classification approaches. Obviously, when there is only a small number of labeled graphs available, supervised approaches cannot work well due to two reasons: (1) During the subgraph features mining procedure, supervised feature selection approaches for graph classification need to employ evaluation criteria to select discriminative subgraph features based on labeled graphs. However, when the labeled graphs are too few, the usefulness of the selected subgraph features can be weak, and thus detri-

ment to the classification performances. (2) During the subgraph feature mining procedure, most supervised graph classification approaches require a branch-and-bound search to avoid exhaustive enumeration of all subgraphs in a dataset. However, when there are not enough labeled graphs, the pruning ability of the upper-bound based on labeled graphs can be poor, thus making it infeasible to find discriminative subgraph features within a reasonable amount of time.

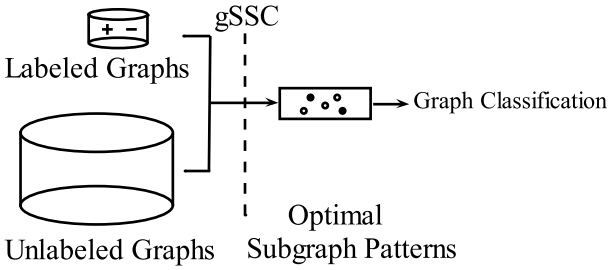
In this chapter, we introduce a novel framework to the above problems by mining subgraph features using both labeled and unlabeled graphs. Our framework is illustrated in 12(b). Different from existing supervised feature selection methods for graph classification, our approach, called gSSC, can utilize both labeled and unlabeled graphs to find optimal subgraph features for graph classification. We first derive a feature evaluation criterion, named gSemi, based upon a given graph dataset with both labeled and unlabeled graphs. Then we propose a branch-and-bound algorithm to efficiently search for optimal subgraph features by deriving an upper-bound of gSemi and pruning the subgraph search space using labeled and unlabeled graphs. In order to evaluate our model, we perform comprehensive experiments on real-world graph classification tasks. The experiments demonstrate that the proposed semi-supervised feature selection method for graph classification outperforms supervised approaches and is very efficient by pruning the subgraph search space using both labeled and unlabeled graphs.

The rest of the chapter is organized as follows. We start by a brief review on related works of graph feature selection and semi-supervised feature selection in Section 3.2. We then introduce the preliminary concepts, give the problem analysis and present the gSemi criterion in

Section 3.3. In Section 3.4, we derive an upper-bound of gSemi and propose the gSSC method. Then Section 3.5 reports the experiment results on real-world graph classification tasks. In Section 3.6, we conclude the chapter.



(a) Supervised Feature Selection Process for Graph Classification



(b) gSSC Semi-Supervised Feature Selection Process for Graph Classification

Figure 12. Different Feature Selection Frameworks for Graph Classification

3.2 Related Work

To the best of our knowledge, this chapter is the first work on semi-supervised feature selection problem for graph classification. Some research works have been done in related areas.

Dimensionality reduction and feature selection in vector spaces have also been studied. Several recent works use pairwise constraints as weak supervision for dimensionality reduction, *i.e.* *must-link* constraints (40) (pairs of instances with the same class) and *cannot-link* constraints (41) (pairs of intstances with different classes). Feature selection methods in vector spaces using both labeled and unlabeled instances have also been proposed (42; 43), which select useful features within a pre-defined feature set. These methods assume that a set of candidate features is given before the feature selection. However, conventional semi-supervised feature selection approaches cannot be directly applied to graph data, because it is usually infeasible to generate all the subgraph features of a graph dataset before feature selection. The number of subgraphs is usually too large to be fully generated, since it grows exponentially with the graph size. Instead, our proposed semi-supervised feature selection for graph data works in a progressive way: the semi-supervised feature selection is integrated to the subgraph feature generation, which can skip most of the bad subgraph features without even generating them.

3.3 Problem Formulation

In this section, we formulate the semi-supervised feature selection problem for graph classification based on subgraph features.

3.3.1 Semi-Supervised Feature Selection

Before presenting the semi-supervised feature selection model for graph classification, we first introduce the notations that will be used throughout this chapter. Let $\mathcal{D} = \{G_1, \cdots, G_n\}$ denote the entire graph dataset, which consists of n graph objects, represented as *connected graphs*. The data set includes both labeled and unlabeled graphs. We assume that the first l

graphs within $\mathcal{D}$ are labeled by $\{y_1, \cdots, y_l\}$, where $y_i \in \{-1, +1\}$ denotes the binary class label assigned to G_i. For convenience, we also denote the labeled graph dataset by $\mathcal{D}_l = \{G_1, \cdots, G_l\}$, and the unlabeled graph dataset as $\mathcal{D}_u = \{G_{l+1}, \cdots, G_n\}$, $\mathcal{D} = \mathcal{D}_l \cup \mathcal{D}_u$.

Definition 5 (Connected Graph) *A graph is represented as $G = (\mathcal{V}, E, \mathcal{L})$, where $\mathcal{V}$ is a set of vertices $\mathcal{V} = \{v_1, \cdots, v_{n_v}\}$, $E \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, $\mathcal{L}$ is the set of symbols for the vertices and the edges. A connected graph is a graph such that there is a path between any pair of vertices.*

Definition 6 (Subgraph) *Let $G' = (\mathcal{V}', E', \mathcal{L}')$ and $G = (\mathcal{V}, E, \mathcal{L})$ be connected graphs. G' is a subgraph of $G (G' \subseteq G)$ iff: (1) $\mathcal{V}' \subseteq \mathcal{V}$, (2) $E' \subseteq E$, (3) $\mathcal{L}' \subseteq \mathcal{L}$. If G' is a subgraph of G, then G is a supergraph of G'.*

In this chapter, we adopt the idea of subgraph-based graph classification approaches, which assume that each graph object G_i is represented as a feature vector $\boldsymbol{x}_i = [x_i^1, \cdots, x_i^m]^\top$ corresponding to a set of subgraph patterns $\{g_1, \cdots, g_m\}$. Denote x_i^k as the binary feature associated with the subgraph pattern g_k. $x_i^k = 1$ iff g_k is a subgraph of G_i ($g_k \subseteq G_i$), otherwise $x_i^k = 0$.

The key issue of semi-supervised feature selection for graph classification is how to find the most informative subgraph patterns from a limited number of labeled graphs and a large number of unlabeled graphs. So, in this chapter, the studied research problem can be described as follow: in order to train an effective graph classifier, how to efficiently find a set of optimal subgraph features from both labeled and unlabeled graphs?

Mining the optimal subgraph features from both labeled and unlabeled graphs is a non-trivial task due to the following problems:

(P1) How to properly evaluate the usefulness of a set of subgraph features based upon both labeled and unlabeled graphs?

(P2) How to find the optimal subgraph features within a reasonable amount of time by avoiding the exhaustive enumeration? The subgraph feature space of graph objects is usually too large, because the number of subgraphs grows exponentially with the size of the graphs. It is infeasible to completely enumerate all the subgraph features for a given graph dataset.

In the following sections, we will first introduce the optimization framework for selecting informative subgraph features from labeled and unlabeled graphs. Next we will describe our subgraph mining strategy using the evaluation criteria derived from the optimization solution.

3.3.2 Optimization Framework

We first address the problem (P1) discussed in Section 3.3.1 by defining the subgraph feature selection as an optimization problem. Our target is to find an optimal set of subgraph features from both labeled and unlabeled graphs. Formally, let us introduce the following notations:

- $\mathcal{S} = \{g_1, g_2, \cdots, g_m\}$: the given set of all the subgraph features, which are used to predict class membership of graph instances. Usually there is only a subset of the subgraph features $\mathcal{T} \subseteq \mathcal{S}$ relevant to the graph classification task.
- $\mathcal{T}^*$: the optimal set of subgraph features $\mathcal{T}^* \subseteq \mathcal{S}$.
- $J(\mathcal{T})$: an evaluation criterion to estimate the usefulness of subgraph feature subset $\mathcal{T}$.

- X: the matrix consisting binary feature vectors using $\mathcal{S}$ to represent the graph dataset $\{G_1, G_2, \cdots, G_n\}$. $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n] = [\boldsymbol{f}_1, \boldsymbol{f}_2, \cdots, \boldsymbol{f}_m]^\top \in \{0,1\}^{m \times n}$, where $X = [X_{ij}]^{m \times n}$, $X_{ij} = 1$ iff $g_i \subseteq G_j$. The first l graphs are labeled as $y_1, \cdots, y_l$.
- $\mathcal{C}$ and $\mathcal{M}$: $\mathcal{C} = \{(i,j)|y_i y_j = -1\}$ denotes the *cannot-link* pairwise constraint sets among labeled graphs. $\mathcal{M} = \{(i,j)|y_i y_j = 1\}$ denotes the *must-link* pairwise constraint sets among labeled graphs.

We propose the following general optimization framework to select optimal subgraph feature set:

$$\mathcal{T}^* = \underset{\mathcal{T} \subseteq \mathcal{S}}{\operatorname{argmax}} J(\mathcal{T}) \qquad \text{s.t. } |\mathcal{T}| \leq t, \tag{3.1}$$

where $|\cdot|$ denotes the size of the feature set and t is the maximum number of feature selected. The objective function in Equation 3.1 has two components: the evaluation criterion $J(\mathcal{T})$ and the subgraph features of graphs $\mathcal{S}$.

We assume that the optimal subgraph features set should have the following properties: (a) *cannot-link*: labeled graphs in different classes should be far away from each other; (b) *must-link*: labeled graphs in the same class should be close to each other; (c) *separability*: unlabeled graphs should be able to be separated from each other. Intuitively, (a) and (b) only consider the constraints from labeled graphs, and tend to select the most discriminative subgraph features based on the graph labels. They are similar to the LDA (44) criterion. Note (c) incorporates the distribution of unlabeled graphs, and tends to select the subgraph features that can separate graphs far from each other. It is similar to the PCA's assumption, which is expressed as the

average squared distance between unlabeled samples. An opposite example for property (c) is: The subgraph features that are too rare or too frequent in the dataset are not useful at all, because unlabeled graphs cannot be separated from each other using these subgraph features. Similar assumptions have also been used by previous works on dimensionality reduction in vector spaces (42).

Based upon the above properties, we derive an evaluation criterion $J(\mathcal{T})$ as follow:

$$\begin{aligned} J(\mathcal{T}) = \quad & \frac{\alpha}{2|\mathcal{C}|} \sum_{y_i y_j = -1} (D_{\mathcal{T}} \boldsymbol{x}_i - D_{\mathcal{T}} \boldsymbol{x}_j)^2 \\ & - \frac{\beta}{2|\mathcal{M}|} \sum_{y_i y_j = 1} (D_{\mathcal{T}} \boldsymbol{x}_i - D_{\mathcal{T}} \boldsymbol{x}_j)^2 \\ & + \frac{1}{2|\mathcal{D}_u|^2} \sum_{G_i, G_j \in \mathcal{D}_u} (D_{\mathcal{T}} \boldsymbol{x}_i - D_{\mathcal{T}} \boldsymbol{x}_j)^2 \end{aligned} \tag{3.2}$$

where $D_{\mathcal{T}} = \text{diag}(\boldsymbol{d}(\mathcal{T}))$ is a diagonal matrix indicating which features are selected into feature set $\mathcal{T}$ from $\mathcal{S}$, $d(\mathcal{T})_i = I(g_i \in \mathcal{T})$. α, β are two parameters, which control the weights of the three types of constraints. Different settings of α and β can refer to different scenarios, and reflect different beliefs we have for the problem. A discussion on the parameter setting will be presented analytically in Section 3.4.4 and empirically in Section 3.5.4.

By defining a matrix $W = [W_{ij}]^{n \times n}$ as

$$W_{ij} = \begin{cases} \frac{\alpha}{|\mathcal{C}|} & \text{if } y_i y_j = -1 \\ -\frac{\beta}{|\mathcal{M}|} & \text{if } y_i y_j = 1 \\ \frac{1}{|\mathcal{D}_u|^2} & \text{if } G_i, G_j \in \mathcal{D}_u \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

we can rewrite the $J(\mathcal{T})$ in Equation 3.2 as follow:

$$\begin{aligned} J(\mathcal{T}) = & \frac{1}{2} \sum_{i,j} \left(D_{\mathcal{T}} \boldsymbol{x}_i - D_{\mathcal{T}} \boldsymbol{x}_j \right)^2 W_{ij} \\ = & \operatorname{tr}(D_{\mathcal{T}}{}^{\top} X \left(D - W \right) X^{\top} D_{\mathcal{T}}) \\ = & \operatorname{tr}(D_{\mathcal{T}}{}^{\top} X L X^{\top} D_{\mathcal{T}}) \\ = & \sum_{g_k \in \mathcal{T}} (\boldsymbol{f}_k{}^{\top} L \boldsymbol{f}_k) \end{aligned} \tag{3.4}$$

where $\operatorname{tr}(\cdot)$ is the trace of a matrix, D is a diagonal matrix whose entries are column sums of W, *i.e.* $D_{ii} = \sum_j W_{ij}$. $L = D - W$ is a Laplacian matrix.

By denoting function $h(g_k, L) = \boldsymbol{f}_k{}^{\top} L \boldsymbol{f}_k$, the optimization in Equation 3.1 can be written as

$$\begin{aligned} \max_{\mathcal{T}} \quad & \sum_{g_k \in \mathcal{T}} h(g_k, L) \\ \text{s.t.} \quad & \mathcal{T} \subseteq \mathcal{S}, |\mathcal{T}| \leq t \end{aligned} \tag{3.5}$$

Definition 7 (gSemi) *Let $\mathcal{D} = \{G_1, \cdots, G_n\}$ denote a graph dataset, with first l graphs labeled as $y_1, \cdots, y_l$. Suppose W is a matrix defined as Equation 3.3. L is a Laplacian matrix defined as $L = D - W$, where D is a diagonal matrix, $D_{ii} = \sum_j W_{ij}$. We define a quality criterion q called gSemi, for a subgraph feature g as*

$$q(g) = h(g, L) = \boldsymbol{f}_g{}^\top L \boldsymbol{f}_g \tag{3.6}$$

where $\boldsymbol{f}_g = [f_g^{(1)}, \cdots, f_g^{(n)}]^\top \in \{0,1\}^n$ is the indicator vector for subgraph feature g, $f_g^{(i)} = 1$ iff $g \subseteq G_i$ $(i = 1, 2, \cdots, n)$. Since the Laplacian matrix L is positive semi-definite, for any subgraph pattern g, $q(g) \geq 0$.

The optimal solution to the problem in Equation 3.5 can be found by using gSemi to make feature selection on a set of subgraphs $\mathcal{S}$. Suppose the gSemi values for all subgraphs are denoted as $q(g_1) \geq q(g_2) \geq \cdots \geq q(g_m)$ in sorted order. Then the optimal solution to the optimization problem in Equation 3.5 is:

$$\mathcal{T}^* = \{g_i | i \leq t\}. \tag{3.7}$$

3.4 gSSC

In this section, we address the problem (P2) discussed in Section 3.3.1 by proposing an efficient method to find the optimal set of subgraphs features from a dataset with both labeled and unlabeled graphs.

The straightforward method is the exhaustive enumeration: We first enumerate all subgraph patterns in the graph dataset, and then calculate the gSemi values for all subgraph patterns. This method is usually impractical, because the number of subgraphs grows exponentially with the size of the graphs. Inspired by recent graph classification approaches, *e.g.* (13), which put their evaluation criteria into the subgraph pattern mining process and develop constraints to prune search spaces, we take a similar approach by deriving a different constraint from both labeled and unlabeled graphs. In order to avoid the exhaustive search, we proposed a branch-and-bound algorithm, named gSSC, which is summarized as follow: a) Adopt a canonical search space where all the subgraph patterns can be enumerated. b) Search through the space, and find the optimal subgraph features by gSemi. c) Propose an upper bound of gSemi and prune the search space. Details with these three steps will be described in the next subsections.

3.4.1 Subgraph Mining

In this chapter, we adopted a depth first search algorithm, gSpan proposed by Yan et al(28), to enumerate all subgraphs from a graph dataset. The key idea of gSpan(28) is that, instead of enumerating subgraphs and testing for isomorphism, they first build a lexicographic order of all the edges of a graph, and then map each graph to an unique minimum DFS code as its canonical label. The minimum DFS codes of two graphs are equivalent iff they are isomorphic. Details can be found in (28). Based on this lexicographic order, a depth-first search (DFS) strategy is used to efficiently search through all the subgraphs in a DFS code tree. By a depth-first search through the DFS code tree's nodes, we can enumerate all the subgraphs of a graph in their

DFS codes' order. And the nodes with non-minimum DFS codes can be directly pruned in the tree, which saves us from performing an explicit isomorphic test among the subgraphs.

3.4.2 Upper Bound of gSemi

By adopting gSpan's DFS Code Tree, we can efficiently enumerate all the subgraph patterns of a graph dataset in a canonical search space. We now derive an upper bound for the gSemi value which can be used to prune the subgraph search space. A convenient method to compute a upper-bound on gSemi value is given as follow:

Theorem 2 (Upper bound of gSemi) *Given any two subgraphs $g, g' \in \mathcal{S}$, g' is a supergraph of g ($g' \supseteq g$). The gSemi value of g' ($q(g')$) is bounded by $\hat{q}(g)$ (i.e., $q(g') \leq \hat{q}(g)$). $\hat{q}(g)$ is defined as follow:*

$$\hat{q}(g) \triangleq \boldsymbol{f}_g{}^\top \hat{L} \boldsymbol{f}_g \tag{3.8}$$

where the matrix $\hat{L}$ is defined as $\hat{L}_{ij} \triangleq \max(0, L_{ij})$. $\boldsymbol{f}_g = \{I(g \subseteq G_i)\}_{i=1}^n \in \{0,1\}^n$ is a vector indicating which graphs in a graph dataset $\{G_1, \cdots, G_n\}$ contain the subgraph g, $I(\cdot)$ is the indicator function. Suppose the gSemi value of g is $q(g) = \boldsymbol{f}_g{}^\top L \boldsymbol{f}_g$.

Proof 2

$$q\left(g'\right) = \boldsymbol{f}_{g'}{}^\top L \boldsymbol{f}_{g'} = \sum_{i,j:G_i,G_j \in \mathcal{G}(g')} L_{ij}$$

where $\mathcal{G}(g') \triangleq \{G_i | g' \subseteq G_i, 1 \leq i \leq n\}$. *Since* g' *is the supergraph of* g ($g' \supseteq g$), *according to anti-monotonic property, we have* $\mathcal{G}(g') \subseteq \mathcal{G}(g)$. *Also* $\hat{L}_{ij} \triangleq \max(0, L_{ij})$, *we have* $\hat{L}_{ij} \geq L_{ij}$ *and* $\hat{L}_{ij} \geq 0$. *So,*

$$q\left(g'\right) = \sum_{i,j:G_i,G_j \in \mathcal{G}(g')} L_{ij} \leq \sum_{i,j:G_i,G_j \in \mathcal{G}(g')} \hat{L}_{ij}$$

$$\leq \sum_{i,j:G_i,G_j \in \mathcal{G}(g)} \hat{L}_{ij} = \hat{q}\left(g\right)$$

Thus, for any $g' \supseteq g$, $q(g') \leq \hat{q}(g)$.

3.4.3 Pruning Search Space

We can now utilize the upper bound to efficiently prune the DFS Code Tree with a branch-and-bound method. During the depth-first search through the DFS Code Tree, we always maintain the temporally suboptimal gSemi value (denoted by θ) among all the gSemi values calculated before. If $\hat{q}(g) < \theta$, the gSemi value of any supergraph g' of g ($g' \supseteq g$) is no greater than θ. Thus, we can safely prune the subtree from g in the search space. If $\hat{q}(g) \geq \theta$, we cannot prune this space since there might exist a supergraph $g' \supseteq g$ that $q(g') \geq \theta$.

The algorithm gSSC is summarized in Figure 13. We initialize a set of selected subgraphs $\mathcal{T}$ as an empty set. In order to speed up the mining process, we can prune the search space from gSpan by always maintaining the currently top-t best subgraphs according to q. During the course of mining, whenever we reach a subgraph g with $\hat{q}(g) \leq \min_{g_i \in \mathcal{T}} q(g_i)$, we can prune the branches originating from g. This is because for any supergraph $g' \supseteq g$ we have $q(g') \leq \hat{q}(g)$, according to the bound defined in Equation 3.8. As long as the resulting subgraph g can improve

$\mathcal{T}$ = gSSC($\mathcal{D}$, $\boldsymbol{y}_l$, min_sup, t)

Input:

$\mathcal{D}$: Graph data set $\{G_1, \cdots, G_n\}$

$\boldsymbol{y}_l$: The first l graphs' labels, where $\boldsymbol{y}_l = [y_1, \cdots, y_l]^\top$

min_sup : Minimum support threshold

t : number of subgraph feature selected

Process:

```
1  T = ∅, θ = 0;
2  Recursively visit the DFS Code Tree in gSpan:
3    g = currently visited subgraph in DFS Code
     Tree
4    if |T| < t, then
5      T = T ∪ {g};
6    else if q(g) > min_{g'∈T} q(g'), then
7      g_min = argmin_{g'∈T} q(g') and T = T/g_min;
8      T = T ∪ {g} and θ = q(g_min);
9    if q̂(g) ≥ θ and freq(g) ≥ min_sup, then
10     Depth-first search the subtree rooted from
       node g;
11 return T;
```

Output:

$\mathcal{T}$: Set of optimal subgraph features

Figure 13. The gSSC algorithm

the gSemi value of any subgraphs $g_i \in \mathcal{T}$, it is accepted into $\mathcal{T}$ and the least best subgraph is dropped off from $\mathcal{T}$. And then we start searching for the next subgraph in the DFS Code Tree.

We further note that in our experiments among almost all datasets gSemi provides such a bound that we can even omit the support threshold min_sup and still find a set of optimal subgraphs within a reasonable time cost.

3.4.4 Discussion

In this section we show the connection between our framework and various application scenarios of graph classification.

Parameter Setting: There are two parameters in the objective function: α and β, which represent the weights of different constraints based on both labeled and unlabeled graphs. Different settings of these parameters fit the optimization to different scenarios of graph classification:

- $\alpha \neq 0$, $\beta = 0$. In this case, we only consider the *cannot-link* constraints and unlabeled graph's *separability* in subgraph feature selection. No *must-link* constraint is considered, *i.e.* labeled graphs within the same classes are not necessarily close together. α controls how much we assume labeled graphs within different classes should be far from each other. This setting of parameters is useful when there is a large diversity within graphs from the same class. For example, drug molecules that have the same toxicology activities on one animal can have very different structures. Furthermore, if $\alpha = +\infty$, we only trust the cannot-link constraints. This reduce the problem into a supervised feature selection task.
- $\alpha = 0$, $\beta \neq 0$. In this setting of parameter, we only consider the *must-link* constraints and unlabeled graph's *separability* in subgraph feature selection. The larger β is, the more we trust the must-link constraints in feature selection. No *cannot-link* constraint is considered, *i.e.* labeled graphs in different classes are not necessarily far from each other.

- $\alpha = 0$, $\beta = 0$. In this case, we don't trust label constraints. Only unlabeled graph's *separability* is considered in subgraph feature selection. This reduce the problem into an unsupervised feature selection task for the unlabeled graph data.
- $\alpha \neq 0$, $\beta \neq 0$. In this case, we consider all constraints (must-link, cannot-link, unlabeled separability) with different weights. This setting is a typical setting for semi-supervised feature selection, where we need to consider both labeled and unlabeled graphs. The smaller the values of α and β, the more we trust the separability constraints from unlabeled graphs.

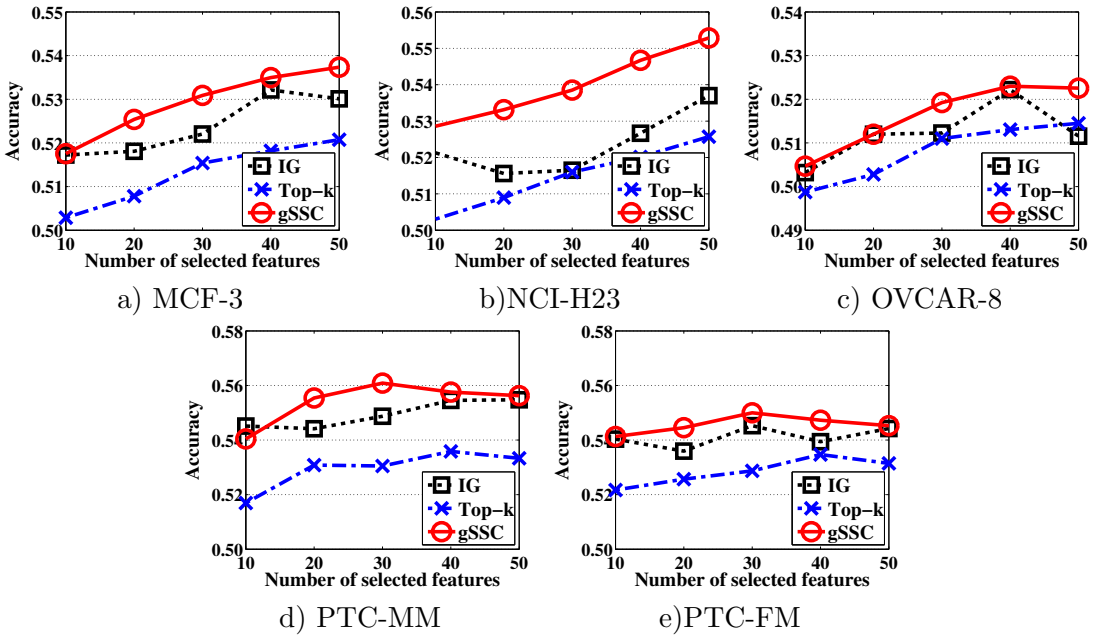


Figure 14. Classification accuracy with different number of features. (#label=30)

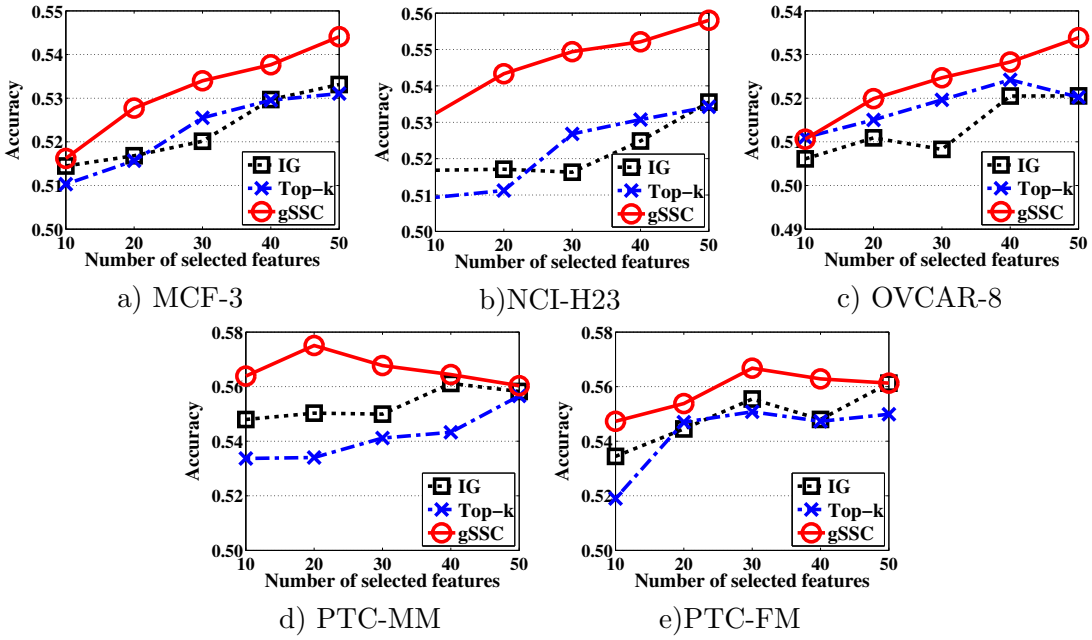


Figure 15. Classification accuracy with different number of features. (#label=50)

3.5 Experiments

In this section, we conduct extensive experiments to examine the effectiveness and efficiency of gSSC in semi-supervised feature selection for graph classification.

3.5.1 Experimental Setup

Data Collections: In order to evaluate the performances of our semi-supervised feature selection approach for graph classification, we tested our algorithm on five real-world graph classification datasets including the following tasks: (Summarized in Table V)

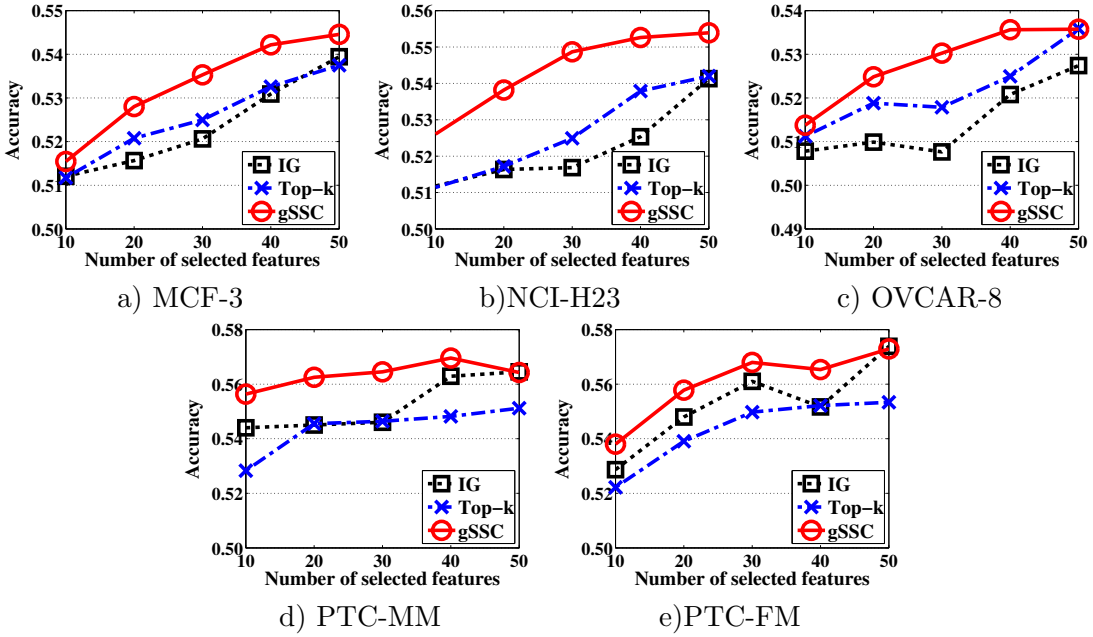


Figure 16. Classification accuracy with different number of features. (#label=70)

1) Anti-cancer activity prediction: The first three benchmark datasets are collect from PubChem Website[1]. The task is to classify chemical compounds' anti-cancer activities on three types of cancers, *i.e.* breast, lung and ovarian. The datasets consist information on the biological activities of small molecules, containing anti-cancer activity records of more then 10,000 chemical compounds against the three types of cancers. Each chemical compound is represented as a graph. We collected 3 graph datasets with *active* and *inactive* labels from PubChem Website. The original datasets are unbalanced, where the

[1]http://pubchem.ncbi.nlm.nih.gov

active class is around 5%. We randomly sample 500 inactive compounds and 500 active compounds from each dataset for performance evaluation.

2) Toxicology prediction (PTC): The last two benchmark datasets are collected from PTC datasets[1] (38). The task is to classify chemical compounds' carcinogenicity on two animal models, *i.e.* MM (Male Mouse) and FM (Female Mouse). The datasets consist carcinogenicity records of more than 300 chemical compounds. Each chemical compound is assigned with carcinogenicity labels for these animal models. On each animal model the carcinogenicity label is one of {CE, SE, P, E, EE, IS, NE, N}. We assume {CE, SE, P} as 'positive' labels, and {NE, N} as 'negative', which is the same setting as (39; 21). Each chemical compound is represented as a graph with an average of 25.7 vertices.

---

[1]http://www.predictive-toxicology.org/ptc/

TABLE V

SUMMARY OF EXPERIMENTAL DATASETS. "POS%" DENOTES THE AVERAGE PERCENTAGE OF POSTIVE GRAPHS IN EACH DATASET.

Name	#Graph	Pos%	Details
MCF-7	27784	8.19	Breast Cancer
NCI-H23	40460	5.06	Lung Cancer
OVCAR-8	40626	5.08	Ovarian Cancer
PTC-MM	336	41.0	Male Mice Toxicology
PTC-FM	349	38.4	Female Mice Toxicology

Comparing Methods: In order to demonstrate the effectiveness of our semi-supervised features selection approach for graph classification, we compare our methods with two baseline methods, including a supervised feature selection approach and an unsupervised approach.

The compared methods are summarized as follows:

- Semi-Supervised **(gSSC)**: The proposed semi-supervised feature selection method for graph classification. We first use gSSC to find a set of subgraph features. The parameters in gSSC are set to $\alpha = \beta = 1$ unless otherwise specified.
- Supervised **(IG)**: We compare with a supervised feature selection method for graph classification. In this approach, a set of frequent subgraphs within labeled graphs are first mined. Then a supervised feature selection based upon Information Gain (IG), an entropy based measure, is used to select a subset of discriminative features from frequent subgraphs.
- Unsupervised **(Top-k)**: We also compare with an unsupervised feature selection method. In this approach, the evaluation criterion for subgraph feature selection is based upon frequency. The top-k frequent subgraph features in labeled graphs are selected.

All experiments are conducted on machines with 4 GB RAM and Intel Xeon^{TM}Quad-Core CPUs of 2.40 GHz.

3.5.2 Performances on Graph Classification

In our experiments, the labeled training graphs are randomly sampled from each datasets. All the remaining graphs are used as unlabeled testing graphs. The results are average of over

30 runs of randomly sampled graph dataset. After the subgraph feature sets are selected by each method, the nearest neighbor (1-NN) classifier is used for classification.

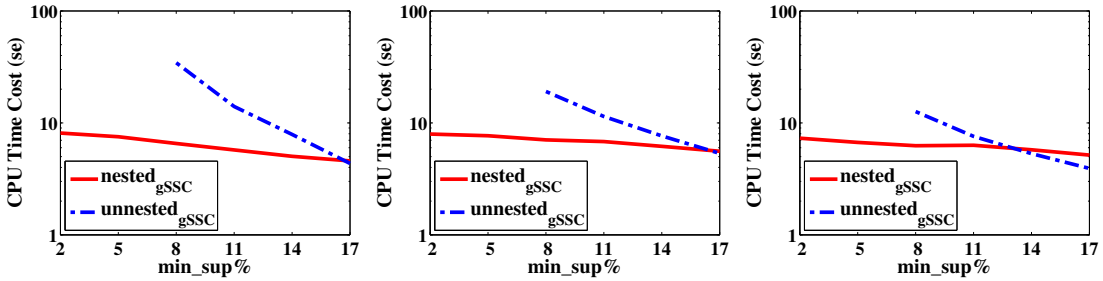
The result of the feature selection methods with different number of labeled training graphs are displayed in Figure 14 (# labeled graphs =30), Figure 15 (# labeled graphs =50) and Figure 16 (# labeled graphs =70). We show the number of selected subgraphs t among frequent subgraphs ($min\_sup = 10\%$), together with classification accuracy as the evaluation metric.

In all these datasets, our semi-supervised feature selection algorithm (gSSC) outperform the supervised approach (IG). gSSC can achieve a good performances with a few labeled training graphs together with a large amount of unlabeled graphs. Although the performance of IG improves with a larger number of features, the IG cannot reach the best performance achievable by gSSC. These results support our first intuition that semi-supervised feature selection methods based on gSemi can boost the performance of graph classification with large amount of unlabeled graphs.

We further observe that gSSC's performances are better than our second baseline Top-k, *i.e.* unsupervised feature selection approaches without label information. These results support our second intuition that the gSemi evaluation criterion in gSSC can find better subgraph patterns for graph classification than unsupervised top-k frequent subgraph selection approaches.

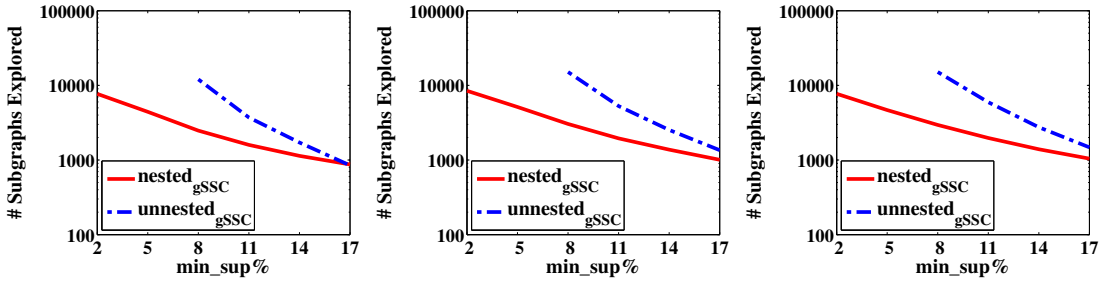
3.5.3 Pruning Search Space

In our second experiment, we evaluated the effectiveness of the upper-bound for gSemi proposed in Section 3.4.2. In this section we compare the runtime performance of two versions of implementation for gSSC: 'nested gSSC' versus 'un-nested gSSC'. The 'nested gSSC' denotes



a) MCF-3 b) NCI-H23 c) OVCAR-8

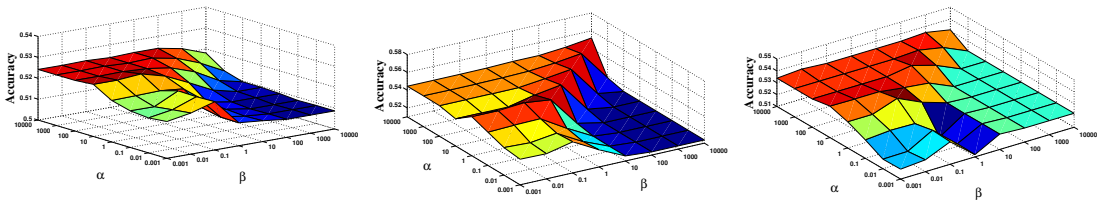
Figure 17. Average CPU time for nested gSSC versus un-nested gSSC with varying min_sup.



a) MCF-3 b) NCI-H23 c) OVCAR-8

Figure 18. Average number subgraph patterns explored during mining for nested gSSC versus un-nested gSSC with varying min_sup.

the proposed method using the upper-bound proposed in Section 3.4.2 to prune the search space of subgraph enumerations; the 'un-nested gSSC' denotes the method without the gSemi's upper-bound pruning, which first uses gSpan to find a set of frequent subgraphs, and then selects the optimal set of subgraphs via gSemi. We run both approaches and record the average CPU time used on feature mining and selection. The result is shown in Figure 17.



a) MCF-3 b) NCI-H23 c) OVCAR-8

Figure 19. Classification accuracy of gSSC with different $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. (#label=50)

In all these datasets, the un-nested gSSC needs to explore increasingly larger subgraph search spaces as we decrease the *min_sup* in the frequent subgraph mining. The size increases exponentially when decreasing *min_sup*. In the MCF-7 dataset, when the *min_sup* get too low ($min\_sup < 8\%$), the subgraph feature enumeration step in un-nested gSSC can run out of the computer memory. However, the nested gSSC's running time does not increase as much, because the gSemi can help pruning the subgraph search space using both labeled and unlabeled graphs. As we can see, the *min_sup* can go to very low value in all datasets for the "nested gSSC".

Figure 18 shows the number of subgraph feature explored in the process of subgraph pattern enumeration. In all datasets, we observe that the number of searched subgraph patterns in nested gSSC is much smaller than that of un-nested gSSC. In our experiments, we further noticed that on most datasets, nested gSSC provides such a strong bound that we may even allow nested gSSC to omit the minimum support threshold *min_sup* and still receive an optimal set of subgraph features within a reasonable time.

3.5.4 Parameter Settings

In our model we can take different weights on constraints from labeled graphs and unlabeled graphs. If we use different setting for the two parameters α and β, we can take the feature selection with different weights for the three types of constraints: must-link, cannot-link and unlabeled separability. α represents how much we weight the cannot-link constraints, and β denotes how much we weight the must-link constraints. The larger α is, the further away the graphs with different classes are separated from each other. The larger β is, the closer the graphs with the same classes are from each other. We test α and β with values among $\{0.001, 0.01, \cdots 10000\}$ separately. The result in Figure 19 shows that the performance of our model using α with large values and β with small values is often better than other settings. The reason is that in these real-world graph classification tasks, graphs in the same class are not always similar with each other, actually graphs can be very different within a same class.

In Figure 19, we find the best parameter setting for MCF-3 dataset is $\alpha = 1$, $\beta = 0.1$ (accuracy = 0.526), and with our default parameter setting ($\alpha = \beta = 1$) the accuracy is 0.523. For NCI-H23 dataset, the best parameter setting is $\alpha = 1$, $\beta = 0.1$ (accuracy= 0.556), and the accuracy with default setting is 0.553. For OVCAR-8 dataset, the best parameter setting is $\alpha = 1$, $\beta = 0.1$ (accuracy= 0.539), and the accuracy with default setting is 0.530. Generally, we can see that the performance of gSSC with default setting ($\alpha = \beta = 1$) is pretty good. If we try to optimize the selection of α and β value, the accuracy improvement relative the two base line schemes will be even bigger.

3.6 Conclusion

In this chapter, we study the problem of semi-supervised feature selection for graph classification. It is significantly more challenging than the conventional setting of supervised feature selection in graph data because of the lack of labeled training graphs. To address this challenge, we propose a feature evaluation criterion, named gSemi, to evaluate subgraph features with both labeled and unlabeled graphs, and derive an upper-bound for gSemi to prune the subgraph search space. Then we propose a branch-and-bound algorithm to efficiently find a set of optimal subgraph feature which is useful for graph classification. Empirical studies on real-world tasks show that our semi-supervised feature selection approach for graph classification outperforms supervised and unsupervised approaches and is very efficient by pruning the subgraph search space using both labeled and unlabeled graphs.

CHAPTER 4

DUAL ACTIVE FEATURE AND SAMPLE SELECTION FOR GRAPH CLASSIFICATION

4.1 Introduction

Conventional approaches on graph classification focus on mining discriminative subgraph features (45) under supervised settings. They assume explicitly or implicitly that a large number of labeled graphs are available. However, in many real-world applications, labeling graph data can be very expensive and time consuming. For example, in molecular medicine, it is very expensive to test the anticancer activity by preclinical studies and clinical trials; in software engineering, human experts have to examine a program flow carefully in order to find software bugs. The labeling cost for graph data can be significantly reduced by training a model that can select the most important graph to query for the label. This setting is also known as *active learning* or *active query selection*. It aims to exploit unlabeled data effectively by selecting important examples to query for labels. Thus, active learning approaches can usually achieve performance comparable to supervised approaches, while using less labeled data. Active learning has been shown to be useful in many real-world applications (46; 47).

Formally, the active learning problem for graph data corresponds to learning a model to select important graphs to obtain class labels. Active learning is particularly challenging in graph data. Conventional active learning approaches estimate the importance of unlabeled

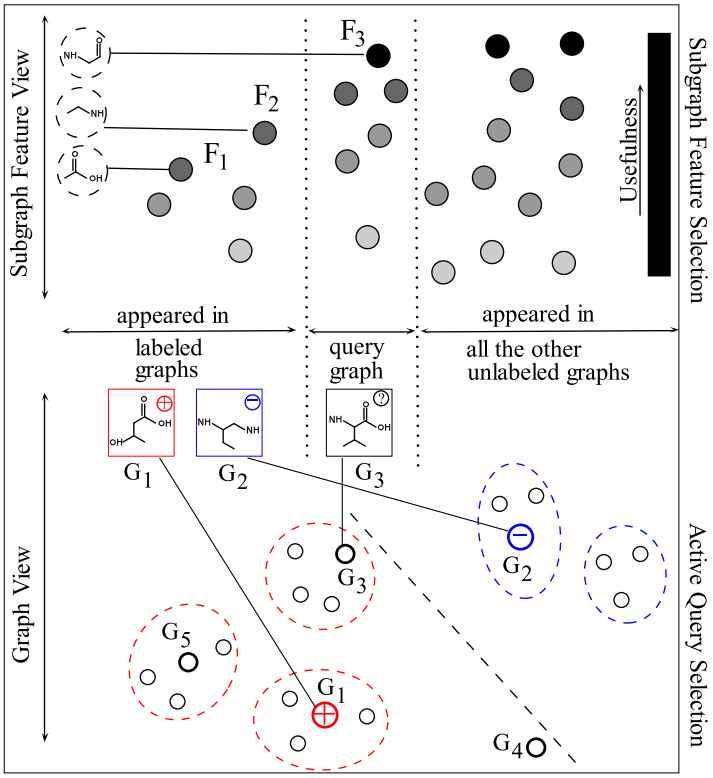


Figure 20. An example of dual active feature and sample selection in graph data.

examples, and assume that all useful features are given. However, in graph data the useful features are not available. Thus, additional steps of subgraph feature mining and selection are required to estimate the usefulness of subgraphs. What makes this problem even more interesting and challenging is that subgraph enumeration and graph isomorphism testing problems are NP-complete. Thus, it is impossible to enumerate all subgraph features and adopt existing approaches for active learning.

In active learning for graph data, the active query selection problem and the subgraph feature selection problem are closely related to each other. The reasons are summarized as follows:

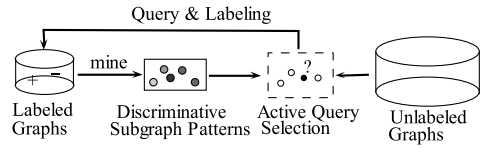
- In active query selection, we need to estimate the importance of each unlabeled graph in order to select the most important graph to query for the label. However, before the most important graph can be determined, we need to find a set of useful subgraph features. Graph data are not directly represented in a meaningful feature space. The performance of the active sample selection directly depends on the quality of the subgraph features mined from the graph dataset. For example, in Figure 20, G_3 is the most important graph which we want to query for the label. G_3 is close to the class boundary like graph G_4. Moreover, G_3 is representative of a cluster of unlabeled graphs. However, the informativeness and representativeness of a graph object depends on which feature set is used. The better the feature set we use, the better we will be able to estimate the importance of the query graphs.

- In the subgraph feature selection problem, we also need to select a set of important subgraph features for the graph classification task. Conventional feature selection approaches for graph data focus on supervised settings (13; 48). The feature evaluation strategies strictly follow the assumption that a large number of labeled graphs are available. However, in the active learning settings, we can only afford to query a small number of unlabeled graphs and obtain their labels. The performance of the feature selection process depends strongly on the quality of the queried graphs in the active sample selection

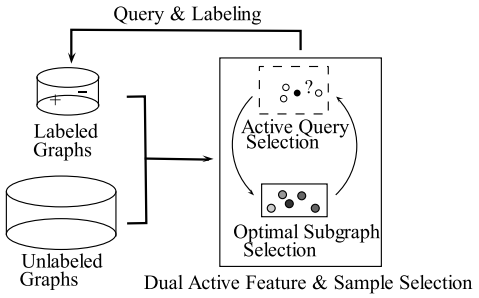
process. For example, in Figure 20, suppose we are given two labeled graphs (G_1 and G_2). Only a small number of the useful subgraph features (F_1 and F_2) appear in the labeled graphs. If we query the graph G_3 that is both representative and informative, we are more likely to find new features like F_3. The process of query selection can assist the process of feature selection in finding useful subgraph features. In other words, the better the query graph we select, the more effectively we can discover the useful subgraph features.

Thus, the active sample selection problem and the subgraph feature selection problem are correlated and should be considered simultaneously. The combined problem is referred to as *dual active feature and sample selection for graph classification*. This problem can be summarized as follows: in order to minimize the labeling cost in graph classification, we need to determine how one can actively select the most important graph to obtain the class label.

To the best of our knowledge, the dual active feature and sample selection for graph classification has not been studied in this context. A straightforward solution to this problem would be the two-stage active learning framework as shown in 21(a). In this framework, the feature selection problem and the active sample selection problem are considered in two separate steps. In the first step, we select a set of subgraph features based upon the labeled graphs. In the second step, we estimate the importance of query graphs based upon the feature set selected in the first step. Obviously, only the subgraph features that appear in the label graphs can be found in the first step. The useful features that only appear in the unlabeled graphs can not



(a) Two-stage active learning in graph data



(b) Dual active feature and sample selection

Figure 21. Different active learning frameworks for graph classification.

be found. Thus, when we only use the features found in the first step, the estimation of the importance of query graphs will not be accurate.

In this chapter we introduce a novel framework for the above problems by exploiting useful subgraph features and optimal query graphs simultaneously. Our framework is illustrated in 21(b). Unlike the two-stage active learning method, the proposed approach, called gActive, can estimate the usefulness of a query graph and effectiveness of subgraph features simultaneously. The gActive method maximizes the dependence between subgraph features and graph labels based upon an active learning framework. Furthermore, we propose a branch-and-bound algorithm to search for optimal features efficiently by pruning the subgraph search space. Empirical

studies on real-world tasks demonstrate that the proposed method can obtain promising results using fewer labeled graphs than alternative approaches.

4.2 Related Work

Active learning aims at reducing the labeling cost by querying the most informative example. Many methods have been proposed based upon different active learning settings. Please see (49) for a detailed survey. Conventional active learning approaches focus on data in vector space. One approach is to query the most informative instance. The active learners select the uncertain instances based upon a single classifier (46; 50) or a committee of classifiers (51; 52; 53). The problems with this approach are that it can be sensitive to outliers or noise, and can not exploit the structures of unlabeled data. The alternative approach is to query the most representative instance. The active learners exploit the structure of unlabeled data using clustering methods (54; 55) or optimal experimental design approaches (56). The major problems are that this approach is unsupervised and can not use the labeled data. Some work has also been done to combine informativeness and representativeness measures to find the optimal query examples (57; 58).

4.3 Problem Formulation

Suppose we are given a graph dataset $\mathcal{D} = \{G_1, \cdots, G_n\}$ that consists of n graphs. $\mathbf{y} = [y_1, \cdots, y_n]^\top$ denotes the vector of labels, where $y_i \in \{+1, 0, -1\}$ is the label of G_i. $y_i = 0$ implies that G_i is unlabeled. Active learning in graph data is the task of selecting one graph G_s from the pool of unlabeled graphs to query for its label. For convenience, we partition the graph dataset into three parts: the labeled graphs $\mathcal{D}_\ell$, the query graph G_s, and the remaining

TABLE VI. Important Notations.

Symbol	Definition
$\mathcal{D} = \{G_1, \cdots, G_n\}$	given graph dataset, G_i denotes the i-th graph in the dataset.
n_ℓ, n_a and n_u	number of labeled graphs, unlabeled graphs including and excluding the query graph in $\mathcal{D}$
$l = \{1, \cdots, n_\ell\}$	index set for labeled graphs in $\mathcal{D}$
s and u	index of selected graph and the index set of the other unlabeled candidate graphs in $\mathcal{D}$.
$a = \{n_\ell + 1, \cdots, n\}$	index set for all unlabeled graphs in the pool including the selected graph. $a = \{s\} \cup u$
$\mathbf{y} = [y_1, \cdots, y_n]^\top$	class label vector for graphs in $\mathcal{D}$, $y_i \in \{+1, -1, 0\}$
$\mathcal{S} = \{g_1, \cdots, g_m\}$	set of all subgraph patterns in the graph dataset $\mathcal{D}$.
$\mathbf{x}_i = [x_i^1, \cdots, x_i^m]^\top$	binary vector for G_i using subgraph features in $\mathcal{S}$, $x_i^k \in \{0, 1\}$ and $x_i^k = 1$ iff $g_k \subseteq G_i$
$\mathbf{f}_i = [f_i^1, \cdots, f_i^n]^\top$	binary vector for subgraph pattern g_i in the $\mathcal{D}$
$X = [X_{ij}]_{(m \times n)}$	matrix of all binary feature vectors in the dataset, $X = [\mathbf{x}_1, \cdots, \mathbf{x}_n] = [\mathbf{f}_1, \cdots, \mathbf{f}_m]^\top$
$\mathcal{T}$	set of selected subgraph patterns, $\mathcal{T} \subset \mathcal{S}$
$\mathbf{K}(\mathcal{T}) = [K_{ij}]_{(n \times n)}$	kernel matrix of all graphs using the selected subgraph features $\mathcal{T}$
$\mathbf{L}(y_s, \mathbf{y}_\ell) = [L_{ij}]_{(n \times n)}$	label kernel matrix of all graphs based upon the class labels
$\mathbf{H} = [H_{ij}]_{(n \times n)}$	centering matrix, $H_{ij} = \delta_{ij} - n^{-1}$. ($\delta_{ij} = 1$ iff $i = j$, otherwise 0)
$D_\mathcal{T}$	an $m \times m$ diagonal matrix indicating which features are selected from $\mathcal{S}$ into $\mathcal{T}$
Π_ℓ, Π_u and Π_s	mapping matrices, $\Pi_\ell \in \{0,1\}^{(n_\ell \times n)}$, $\Pi_s \in \{0,1\}^{(1 \times n)}$, $\Pi_u \in \{0,1\}^{(n_u \times n)}$ and $[\Pi_\ell^\top, \Pi_s^\top, \Pi_u^\top] = \mathbf{I}_n$

unlabeled graphs D_u. $\mathcal{D}_a = \mathcal{D}_u \cup \{G_s\}$ denotes all unlabeled graphs. The vector $\mathbf{y}$ is partitioned as follows:

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_\ell \\ y_s \\ \mathbf{y}_u \end{bmatrix} = \begin{bmatrix} \mathbf{y}_\ell \\ \\ \mathbf{y}_a \end{bmatrix} \quad \text{and} \quad \mathbf{y}_a = \begin{bmatrix} y_s \\ \\ \mathbf{y}_u \end{bmatrix}$$

where $\mathbf{y}_\ell$, y_s and $\mathbf{y}_u$ represent the class labels assigned to the graphs in $\mathcal{D}_\ell$, $\{G_s\}$ and $\mathcal{D}_u$ respectively. We denote the number of labeled graphs by n_ℓ. n_u is the number of unlabeled graphs excluding the query graph, and n_a is the number of all unlabeled graphs. $n_a = n_u + 1$. We assume the first n_ℓ graphs in the dataset $\mathcal{D}$ are labeled.

Definition 8 (Graph) *A graph is represented as $G = (\mathcal{V}, E, \mathcal{L}, l)$. $\mathcal{V}$ is the set of vertices, and $\mathcal{V} = \{v_1, \cdots, v_{n_v}\}$. $E \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. $\mathcal{L}$ is the set of labels for the vertices and edges. $l : \mathcal{V} \cup E \to \mathcal{L}$ is the function assigning labels to the vertices and edges.*

We focus on using subgraph patterns to define the feature space of graph classification. It is assumed that a graph object G_i is represented as a binary vector $\mathbf{x}_i = [x_i^1, \cdots, x_i^m]^\top$ associated

with a set of subgraph patterns $\{g_1, \cdots, g_m\}$. Here, $x_i^k \in \{0,1\}$ is the binary feature of G_i corresponding to the subgraph pattern g_k. $x_i^k = 1$ iff g_k is a subgraph of G_i. Now suppose the full set of subgraph features in the graph dataset $\mathcal{D}$ is $\mathcal{S} = \{g_1, \cdots, g_m\}$, which we use to predict the class labels of the graph objects. The full feature set $\mathcal{S}$ is very large. Only a subset of the features ($\mathcal{T} \subseteq \mathcal{S}$) is relevant to the graph classification task. Let X denote the matrix consisting of the binary feature vectors based on $\mathcal{S}$ to represent the graph dataset $\mathcal{D}$. $X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] = [\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_m]^\top \in \{0,1\}^{m \times n}$, where $X = [X_{ij}]_{m \times n}$, $X_{ij} = 1$ iff $g_i \subseteq G_j$. We briefly summarize the notations used in this chapter in Table VI.

Definition 9 (Subgraph) *Let $G' = (\mathcal{V}', E', \mathcal{L}', l')$ and $G = (\mathcal{V}, E, \mathcal{L}, l)$ be graphs. G' is a subgraph of G ($G' \subseteq G$) iff there exists an injective function $\psi : \mathcal{V}' \to \mathcal{V}$. $\forall v \in \mathcal{V}'$, $l'(v) = l(\psi(v))$. $\forall (u,v) \in E'$, $(\psi(u), \psi(v)) \in E$ and $l'(u,v) = l(\psi(u), \psi(v))$. If G' is a subgraph of G, then G is a supergraph of G'.*

The key issue of *dual active feature and sample selection* is to find the most important query graph and a set of optimal subgraph patterns simultaneously. The problems studied in this chapter are as follows:

1) How can one estimate the importance of a query graph among unlabeled graphs?

2) How can one estimate the usefulness of a set of features for the graph classification task?

3) How can one determine the optimal subgraph features within a reasonable amount of time, and avoid the exhaustive enumeration of all features?

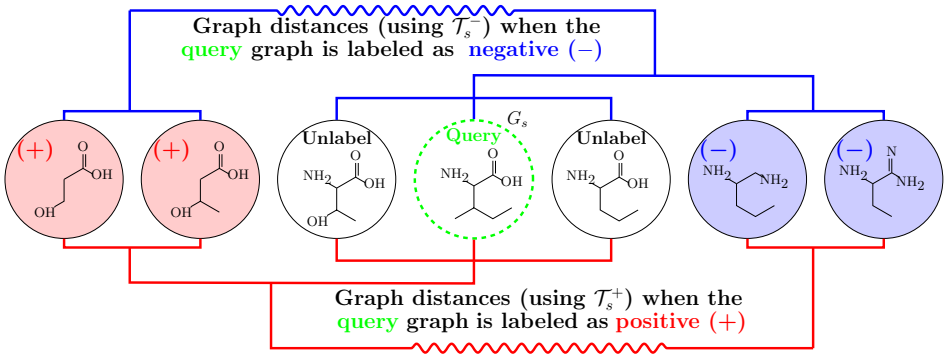


Figure 22. Graph distances using two optimal feature sets ($\mathcal{T}_s^+$ and $\mathcal{T}_s^-$) respectively depending on the label of the query graph.

4.3.1 Optimization Framework

We propose an optimization framework to select the optimal query graph by maximizing the minimum score of an evaluation function.

$$G_s^* = \arg\max_{G_s \in \mathcal{D}_a} \min_{y_s \in \{\pm 1\}} \mathcal{E}(G_s, y_s, \mathcal{D}, \mathbf{y}_\ell) \tag{4.1}$$

where $\mathcal{D}_a$ denotes the pool of all unlabeled graphs. $\mathcal{E}$ denotes the evaluation function for querying a graph G_s in $\mathcal{D}_a$. Because the label of the selected graph G_s can be either 1 or -1, we need to consider both alternatives and select the worst case to maximize. In this max-min view of active learning, it guarantees that the selected graph G_s will lead to a large value for the function $\mathcal{E}(G_s, y_s, \mathcal{D}, \mathbf{y}_\ell)$.

Note that in graph data the useful features are not given. We need to make use of the label information to select a subset of optimal subgraph features. If we know the class label of the

selected query graph G_s, we can select the optimal feature set by maximizing the following function.

$$\mathcal{E}(G_s, y_s, \mathcal{D}, \mathbf{y}_\ell) = \max_{\mathcal{T} \subseteq \mathcal{S}, |\mathcal{T}|=t} J(G_s, y_s, \mathcal{D}, \mathbf{y}_\ell, \mathcal{T}) \tag{4.2}$$

We select a subset of subgraph features $\mathcal{T}$ from $\mathcal{S}$, such that the feature selection evaluation function $J(G_s, y_s, \mathcal{D}, \mathbf{y}_\ell, \mathcal{T})$ is maximized.

4.3.1.1 The Intuition

Now, we show the key ideas of this chapter from the following views.

Query Selection View: from the active query selection view, we assume that the optimal query graph should satisfy the following properties:

(a) Dependence maximization: based upon a feature set $\mathcal{T}$, the query graph G_s should be able to maximize the dependence between features of graphs and labels in a max-min view as in Equation 4.1.

(b) Informative and representative: the selected query graph should be both informative and representative among the pool of unlabeled graphs. The query graph G_s should be close to some of the unlabeled graphs in the dataset, such that G_s is likely to be representative of a group of unlabeled graphs instead of being an outlier. The query graph G_s should also be far from the labeled graphs in $\mathcal{D}_\ell$, such that the label of G_s is unlikely to be redundant.

Feature Selection View: from the subgraph feature selection view, the optimal feature set can be very different depending on which graph we query, and what class label we get

from the domain expert. For example, in Figure 22, we have a graph dataset with seven graph objects. Suppose the selected query graph is G_s. We denote the optimal feature set as $\mathcal{T}_s^+$, when G_s is positive. $\mathcal{T}_s^-$ represents the optimal feature set, when G_s is negative. From the feature selection perspective, the optimal subgraph features should also satisfy the following properties:

(a) Dependence maximization: graphs with the same class labels should have similar subgraph features, and be close to each other; while graphs with different labels should have different features and be far away from each other.

(b) Informative and representative: the query graph G_s should be close to the other unlabeled graphs, and be far from the existing labeled graphs.

4.3.1.2 The Solution

Many criteria can be used to evaluate the dependence between subgraph features and graph labels. In this chapter we adopt the Hilbert-Schmidt Independence Criterion (HSIC) (37) for subgraph evaluation. HSIC measures the dependence between two variables (X, Y) in the kernel space. We briefly review the definition of HSIC. Suppose we have two reproducing kernel Hilbert spaces (RKHS) of functions $\mathcal{G}$ and $\mathcal{F}$. Let a covariance operator be

$$C = \mathrm{E}\left\{[p(X) - \mathrm{E}(p(X))][q(Y) - \mathrm{E}(q(Y))]\right\}, \ \ \forall p \in \mathcal{G}, q \in \mathcal{F}$$

Then the HSIC is defined as the Hilbert-Schmidt norm of the operator C, *i.e.*, $\|C\|_{HS}^2$. Given a data sample, HSIC has an empirical estimator HSIC = tr($\mathbf{K}$ $\mathbf{H}$ $\mathbf{L}$ $\mathbf{H}$). Here, tr$(\cdot)$ is the trace

of a matrix. $\mathbf{H} = [H_{ij}]_{n\times n}$, where $H_{ij} = \delta_{ij} - 1/n$. δ_{ij} is the indicator function. $\delta_{ij} = 1$ iff $i = j$, otherwise $\delta_{ij} = 0$. $\mathbf{K}$ and $\mathbf{L}$ are kernel matrices on the samples.

Based upon the HSIC measure, we propose the following evaluation function for active feature selection:

$$\begin{aligned} J(G_s, y_s, \mathcal{D}, \mathbf{y}_\ell, \mathcal{T}) =& \text{tr}\left[\ \mathbf{K}(\mathcal{T})\ \mathbf{H}\ \mathbf{L}(y_s, \mathbf{y}_\ell)\ \mathbf{H}\ \right] \\ &+ \alpha \frac{\mathbf{1}^\top \mathbf{K}_{u,s}(\mathcal{T})}{n_u} \ - \ \beta \frac{\mathbf{1}^\top \mathbf{K}_{l,s}(\mathcal{T})}{n_\ell} \end{aligned} \tag{4.3}$$

where $\mathbf{K}(\mathcal{T}) = [K_{ij}]_{(n\times n)}$ denotes the kernel matrix of graphs based upon a subgraph feature $\mathcal{T}$. $K_{ij} = \langle D_\mathcal{T}\mathbf{x}_i, D_\mathcal{T}\mathbf{x}_j\rangle$. $D_\mathcal{T} = diag(\mathbf{d}_\mathcal{T})$ is a diagonal matrix, where $\mathbf{d}_\mathcal{T} = [d(\mathcal{T})_i]_{(m\times 1)}$ and $d(\mathcal{T})_i = I(g_i \in \mathcal{T}) \in \{0, 1\}$. $\mathbf{L}(y_s, \mathbf{y}_\ell) = [L_{ij}]_{(n\times n)} = \mathbf{y}\mathbf{y}^\top$ denotes the kernel matrix based on the labels of the graphs, where $\mathbf{y} = [\ \mathbf{y}_\ell^\top, y_s, \mathbf{y}_u^\top\ ]^\top$. $L_{ij} = \langle y_i, y_j\rangle$ is used in our current implementation. Other kernels can also be directly used in this formulation. α and β are two parameters that control the weights of the three terms in the evaluation function. The first term denotes the dependence between the features and labels of the graphs. The second term represents the average similarity between the query graph and the unlabeled graphs. The third term represents the average similarity between the query graph and the labeled graphs.

The evaluation function in Equation 4.3 can be simplified as follows:

$$\begin{aligned} \text{tr}\left(\mathbf{K}(\mathcal{T})\mathbf{H}\mathbf{y}\mathbf{y}^\top\mathbf{H}\right) =& \ \text{tr}\left(X^\top D_\mathcal{T} D_\mathcal{T} X\ \mathbf{H}\ \mathbf{y}\mathbf{y}^\top\ \mathbf{H}\right) \\ =& \ \text{tr}\left(D_\mathcal{T} X\ \mathbf{H}\ \mathbf{y}\mathbf{y}^\top\ \mathbf{H}\ X^\top D_\mathcal{T}\right) \\ =& \ \sum_{g_i\in\mathcal{T}} \left(\mathbf{f}_i^\top \mathbf{H}\ \mathbf{y}\mathbf{y}^\top\ \mathbf{H}\ \mathbf{f}_i\right) \end{aligned}$$

Input:

$\mathcal{D}$: the graph dataset $\{G_1, \cdots, G_n\}$ — t: the maximum number of features.

$\mathbf{y}_\ell$: the vector of class labels for labeled graphs, — min_sup: the minimum frequency.

Initialize:

- Construct the feature evaluation functions $h(g, \mathbf{M})$ and initialize candidate feature lists:
  1. Calculate $2 \times n_a$ matrices using Equation 4.4 by considering each case for G_s and y_s as follows:
     $\mathbf{M}_i^+ = \mathbf{H}\ \mathbf{L}(y_i = +1, \mathbf{y}_\ell)\ \mathbf{H}\ + \frac{\alpha}{n_u}\ \Pi_i \mathbf{1}^\top \Pi_u^\top - \frac{\beta}{n_\ell}\ \Pi_i \mathbf{1}^\top \Pi_\ell^\top$, $(\forall\ i,\ n_\ell < i \leq n)$
     $\mathbf{M}_i^- = \mathbf{H}\ \mathbf{L}(y_i = -1, \mathbf{y}_\ell)\ \mathbf{H}\ + \frac{\alpha}{n_u}\ \Pi_i \mathbf{1}^\top \Pi_u^\top - \frac{\beta}{n_\ell}\ \Pi_i \mathbf{1}^\top \Pi_\ell^\top$, $(\forall\ i,\ n_\ell < i \leq n)$
  2. Initialize $2 \times n_a$ empty lists for candidate subgraph features as follows:
     $\forall\ i\ (n_\ell < i \leq n)$, let $\mathcal{T}_i^+ = \mathcal{T}_i^- = \emptyset$ with maximum size t, and pruning thresholds $\theta_i^+ = \theta_i^- = -\infty$

Recursive Feature Mining:

- Depth-first search the gSpan's code tree and update the feature lists as follows:
  1. Update each of the candidate feature lists using the current subgraph feature g_c:
     $\forall\ i$, if $h(g_c, \mathbf{M}_i^+)$ is larger than the worst feature in $\mathcal{T}_i^+$, replace it and update $\theta_i^+ = \min_{g \in \mathcal{T}_i^+} h(g, \mathbf{M}_i^+)$
     $\forall\ i$, if $h(g_c, \mathbf{M}_i^-)$ is larger than the worst feature in $\mathcal{T}_i^-$, replace it and update $\theta_i^- = \min_{g \in \mathcal{T}_i^-} h(g, \mathbf{M}_i^-)$
  2. Test pruning criteria for the sub-tree rooted from node g as follows:
     if $freq(g_c) < min\_sup$, prune the sub-tree of g_c
     if $\forall\ i\ (n_\ell < i \leq n)$, $\tilde{h}(g_c, \mathbf{M}_i^+) \leq \theta_i^+$ and $\tilde{h}(g_c, \mathbf{M}_i^-) \leq \theta_i^-$, prune the sub-tree of g_c
  3. Recursion: Depth-first search the sub-tree rooted from node g_c

Active Query Selection:

- Select the query graph using Equation 4.5

Output:

G_s: The selected query graph.

$\mathcal{T}_s^+$: the optimal subgraph features if G_s is labeled as a positive graph.

$\mathcal{T}_s^-$: the optimal subgraph features if G_s is labeled as a negative graph.

Figure 23. The gActive algorithm

We can rewrite $J(G_s, y_s, \mathcal{D}, \mathbf{y}_\ell, \mathcal{T})$ in Equation 4.2 as

$$
\begin{aligned}
& J(G_s, y_s, \mathcal{D}, \mathbf{y}_\ell, \mathcal{T}) \\
&= \sum_{g_i \in \mathcal{T}} \mathbf{f}_i^\top \left(\mathbf{H}\ \mathbf{y}\mathbf{y}^\top\ \mathbf{H}\ + \frac{\alpha}{n_u}\ \Pi_s \mathbf{1}^\top \Pi_u^\top - \frac{\beta}{n_\ell}\ \Pi_s \mathbf{1}^\top \Pi_\ell^\top \right) \mathbf{f}_i \\
&= \sum_{g_i \in \mathcal{T}} \mathbf{f}_i^\top\ \mathbf{M}\ \mathbf{f}_i
\end{aligned}
$$

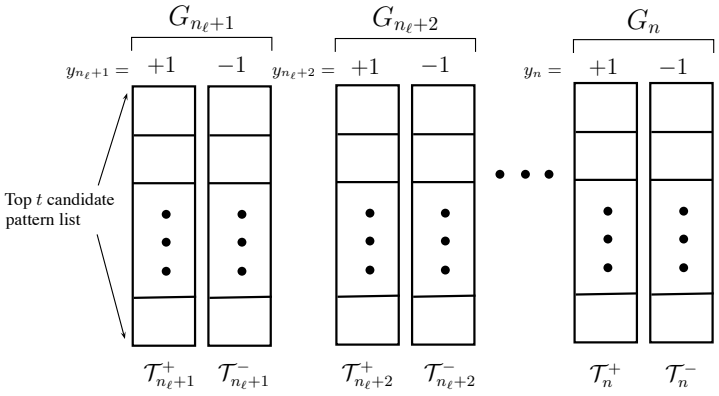


Figure 24. Candidate subgraph pattern lists

where Π_ℓ and Π_u are projection matrices. $X_\ell = X\Pi_\ell$, $X_u = X\Pi_u$ and $\mathbf{x}_s = X\Pi_s$.

$$\mathbf{M} = \mathbf{H}\ \mathbf{y}\mathbf{y}^\top\ \mathbf{H}\ + \frac{\alpha}{n_u}\ \Pi_s\mathbf{1}^\top\Pi_u^\top - \frac{\beta}{n_\ell}\ \Pi_s\mathbf{1}^\top\Pi_\ell^\top \tag{4.4}$$

Given the evaluation function $h(g_i, G_s, y_s, \mathcal{D}, \mathbf{y}_\ell) = \mathbf{f}_i^\top\ \mathbf{M}\ \mathbf{f}_i$, we can define the optimization problem of dual active feature and sample selection as follows:

$$G_s^* = \arg\max_{G_s \in \mathcal{D}_a}\ \min_{y_s \in \{\pm 1\}}\ \max_{\mathcal{T} \subseteq \mathcal{S}, |\mathcal{T}| = t} \sum_{g_i \in \mathcal{T}} h(g_i, G_s, y_s, \mathcal{D}, \mathbf{y}_\ell) \tag{4.5}$$

Definition 10 (gFScore) *Let $\mathcal{D} = \{G_1, \cdots, G_n\}$ denote a graph dataset, with first n_ℓ graphs labeled as $y_1, \cdots, y_{n_\ell}$. Suppose we have a query graph G_s with its potential label y_s. We define a quality criterion $h(g_i, G_s, y_s, \mathcal{D}, \mathbf{y}_\ell) = h(g_i, \mathbf{M}) = \mathbf{f}_i^\top\ \mathbf{M}\ \mathbf{f}_i$, called gFScore, for a subgraph feature g_i. $\mathbf{M}$ is a matrix defined in Equation 4.4.*

The optimal solution to Equation 4.5 can be found by the following method: we go through each unlabeled graph one by one. Suppose G_s is the currently selected graph. We need to go through a positive case ($y_s = 1$) and a negative case ($y_s = -1$). In each case, we mine the top-t best subgraph features according to the gFScore. As shown in Figure 24, we need to mine $2 \times n_a$ optimal feature sets from the graph dataset. $\mathcal{T}_i^+$ denotes the optimal feature set, when the i-th graph is queried and labeled as a positive graph. $\mathcal{T}_i^-$ denotes the optimal feature set, when G_i is queried and labeled as negative. Then, we can directly use Equation 4.5 to find the optimal query graph.

4.3.1.3 Upper Bound of gFScore

Now we address the problem on how to mine the optimal feature sets without exhaustive enumeration of all subgraphs. Note that the number of subgraphs in a graph dataset can be extremely large. It grows at an exponential rate as the size of the graphs increases. Thus, it is computationally intractable to enumerate all subgraphs in the graph dataset.

Some recent graph classification approaches (13; 14; 34) incorporate constraints to prune the search space of gSpan (28). In this chapter we derive a new constraint to prune the search space in the DFS-code tree. A straightforward upper-bound of gFScore is defined as follows:

Theorem 3 (Upper bound of gFScore) *Suppose we have two subgraph patterns $g_i, g_j \in \mathcal{S}$, and g_j is a supergraph of g_i ($g_j \supseteq g_i$). The gFScore value of g_j is bounded by $\widetilde{h}(g_i, \mathbf{M})$, i.e., $h(g_j, \mathbf{M}) \leq \widetilde{h}(g_i, \mathbf{M})$. $\widetilde{h}(g_i, \mathbf{M})$ is defined as follows:*

$$\widetilde{h}(g_i, \mathbf{M}) \triangleq \mathbf{f}_i{}^\top \widetilde{\mathbf{M}} \mathbf{f}_i \tag{4.6}$$

where the matrix $\widetilde{\mathbf{M}}$ is defined as $\widetilde{M}_{pq} \triangleq \max(0, M_{pq})$.

Proof 3

$$h\left(g_j, \mathbf{M}\right) = \mathbf{f}_j{}^{\top}\mathbf{M}\mathbf{f}_j = \sum_{p,q:G_p,G_q \in \mathcal{D}(g_j)} M_{pq} \tag{4.7}$$

where $\mathcal{D}(g_j) \triangleq \{G_k | g_j \subseteq G_k, 1 \leq k \leq n\}$. *Since* g_j *is a supergraph of* g_i ($g_j \supseteq g_i$), *we have* $\mathcal{D}(g_j) \subseteq \mathcal{D}(g_i)$ *according to the anti-monotonic property.* $\widetilde{M}_{pq} \triangleq \max(0, M_{pq})$. *We have* $\widetilde{M}_{pq} \geq M_{pq}$ *and* $\widetilde{M}_{pq} \geq 0$. *Thus,*

$$\begin{aligned} h\left(g_j, \mathbf{M}\right) &= \sum_{p,q:G_p,G_q \in \mathcal{D}(g_j)} M_{pq} \leq \sum_{p,q:G_p,G_q \in \mathcal{D}(g_j)} \widetilde{M}_{pq} \\ &\leq \sum_{p,q:G_p,G_q \in \mathcal{D}(g_i)} \widetilde{M}_{pq} = \widetilde{h}\left(g_i, \mathbf{M}\right) \end{aligned} \tag{4.8}$$

For any $g_j \supseteq g_i$, $h(g_j, \mathbf{M}) \leq \widetilde{h}(g_i, \mathbf{M})$.

We now utilize the upper bound to prune the DFS-code tree in gSpan by the branch-and-bound pruning. The top-t best features are maintained in $2 \times n_a$ candidate lists. Figure 24 shows an example of the candidate lists. During the course of the subgraph pattern mining, we calculate the upper-bound of each subgraph pattern in the search tree. If a subgraph pattern node with its children nodes cannot update any of the candidate feature lists, we can prune the subtree of gSpan rooted from this node. It is guaranteed by the upper-bound that we will not miss any better features for any of the candidate feature lists. Thus, the subgraph feature mining process can be speeded up without loss of performance. The algorithm of gActive is summarized in Figure 23.

4.4 Experiments

Data Collection: in order to evaluate the performance of the proposed approach for graph classification, we tested our algorithm on nine real-world datasets as summarized in Table VII.

1) *Anti-cancer activity prediction (NCI)*: the first eight benchmark datasets are collected from the *PubChem* website[1]. The datasets contain records of anticancer activities for more than 20,000 chemical compounds against eight types of cancer. Each chemical compound is represented as a graph. We collected eight graph data sets with *active* and *inactive* labels from the *PubChem* website. The original datasets are unbalanced, where the percentage of positive compounds is around 5%. We randomly sampled 500 inactive compounds and 500 active compounds from each dataset for performance evaluation.

2) *AIDS anti-virus prediction (HIV)*: the last benchmark dataset is collected from the AIDS anti-viral screening program[2]. The dataset consists of screening records of more than 7700 chemical compounds. Each compound is described by its activity against HIV, which is one of the following categories: confirmed active (CA), confirmed moderately active (CM) and confirmed inactive (CI). We treat CA+CM as the *positive* label, and CI as the *negative* label. This is the same setting used in (59). The original data set is unbalanced, where the percentage of positive compounds is around 3%. We randomly sampled 266 inactive compounds and used the original 266 active compounds for performance evaluation.

Comparative Methods: in order to demonstrate the effectiveness of the proposed approach, we compare our method against four baseline methods. These baseline methods include

---

[1]http://pubchem.ncbi.nlm.nih.gov

[2]http://dtp.nci.nih.gov/

TABLE VII

SUMMARY OF EXPERIMENTAL DATASETS. "# POS" DENOTES THE NUMBER OF ACTIVE GRAPHS IN THE DATASET.

Dataset	# Pos	# Graph	Details
NCI1	2040	40526	Lung Cancer
NCI33	1636	40209	Melanoma
NCI41	1561	27585	Prostate Cancer
NCI47	2011	40447	Central Nerve System
NCI81	1396	40700	Colon Cancer
NCI83	2276	27992	Breast Cancer
NCI123	3112	40152	Leukemia
NCI145	1940	40164	Renal Cancer
AIDS	266	7781	HIV Anti-virus

both supervised and unsupervised feature selection approaches combined with active sample selection approaches. The compared methods are summarized as follows:

- *Dual Active Feature and Sample Selection (gActive)*: the proposed method in this chapter. The default parameter setting for the gActive method is $\alpha = \beta = 10^{-3}$.
- *Supervised Feature Selection + Random Sampling (IG + Random)*: we compare with a supervised feature selection method combined with random sampling. A set of frequent subgraphs within the graph dataset is first mined. Then a supervised feature selection method based upon Information Gain (IG) is used to select a subset of discriminative features from the frequent subgraphs. We randomly select query graphs from the pool. Note that discriminative features are recomputed after each iteration of feature and graph selection.

- *Supervised Feature Selection + Margin (IG + Margin)*: we compare with a two-stage active learning approach. A supervised feature selection method is combined with a margin-based active learning method. In this approach, a set of frequent subgraphs within the graph dataset are first mined. Then we iteratively use information gain to select a subset of discriminative features from the frequent subgraphs. We use a margin-based active learning approach (46) to select informative graphs to query for labels.
- *Unsupervised Feature Selection + Random Sampling (Top-k + Random)*: we compare with an unsupervised feature selection method combined with random sampling. In this approach, we use the top-k frequent subgraph features in the pool dataset. Then we randomly select query graphs from the pool.
- *Unsupervised Feature Selection + Margin (Top-k + Margin)*: we compare with an unsupervised feature selection method combined with a margin-based active learning method. In this approach, we use the top-k most frequent subgraphs in the pool dataset. Then margin-based active learning is used to select informative graphs to query.
- *Unsupervised Feature Selection + Sequential TED (Top-k + TED)*: we compare with an unsupervised feature selection method combined with experimental designs. In this approach, we use the top-k most frequent subgraphs in the pool dataset. Then the sequential transductive experimental design (56) is used to select representative graphs from the pool dataset.

All experiments are conducted on machines with Intel Xeon^{TM} Quad-Core CPUs of 2.27 GHz and 24 GB RAM. LibSVM (60) with the linear kernel is used as the base classifier for all

compared methods. min_sup =10% in the gSpan. The default number of selected features in all compared methods is set as 500.

4.4.1 Performance on Graph Classification

In our experiments, we first randomly sample two labeled graphs, *i.e.*, one positive and one negative graph. They are used as the initial training set. Then we partition the remaining graphs into two groups with equal size: one group is used as the pool dataset; the other group is used as the testing dataset for performance evaluation. In each iteration, the active learner selects one unlabeled graph in the pool dataset and queries the label. Then, the queried graph is put into the training set, and the base classifier is trained to classify the testing graphs. We report the average results of 50 runs on randomly sampled graph datasets.

The results of all compared methods are summarized in Figure 25. We run each of the methods for 50 iterations and compare the learning curves. In all datasets, the proposed dual active feature and sample selection algorithm (gActive) consistently outperforms other baseline methods. Note that in the NCI33 and NCI145 datasets, all the baselines with margin-based active learning (IG + Margin and Top-k + Margin) are unable to achieve better performance than randomized baselines (IG + Random and Top-k + Random). However, our method can achieve substantially better performance than other baselines. This result supports the intuition of this chapter: the feature selection problem and the active query selection problem are correlated, and should be considered simultaneously. Moreover, we notice that in the NCI41 dataset, all the baselines using margin-based active learning can improve the performance over

randomized baselines. In this dataset, the improvements by the proposed gActive method can be even more significant.

4.4.2 Parameter Settings

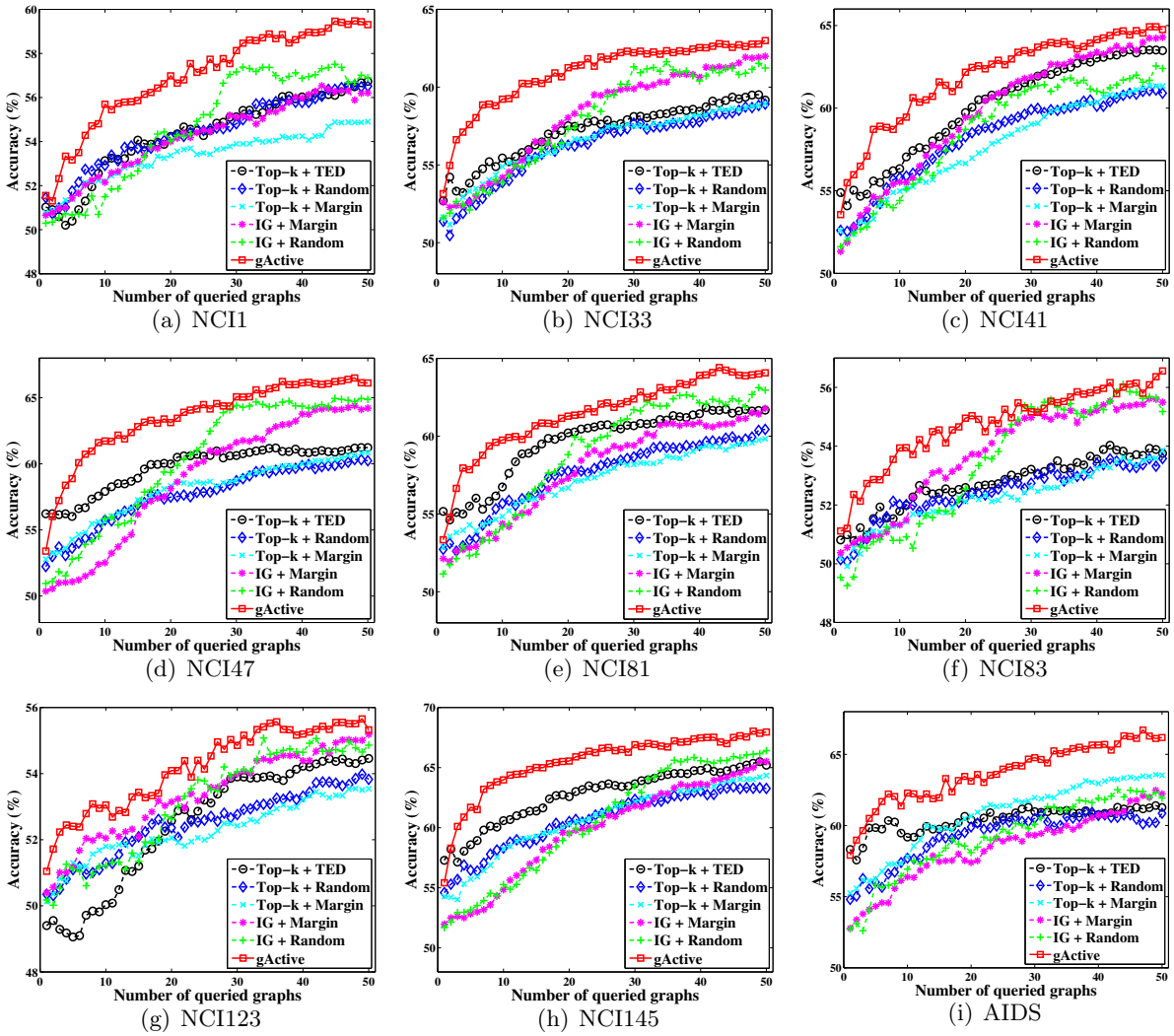
In the proposed model, different weights can be assigned to the three terms in Equation 4.3. If we use different settings for the two parameters, α and β, we can perform the dual active feature and sample selection with different weights for the three constraints: *dependence maximization*, *representativeness* and *informativeness*. To be precise, α represents how much we weight the *representativeness*, and β denotes how much we weight the *informativeness*. The larger the value of α is, the closer the query graph is with other unlabeled graphs. The larger the value of β is, the further away the query graph is from the other labeled graphs. We test α and β with values among {100, 10, 1, 0.1, 0.01, 0.001, 0} separately. The average results for our model in the first 50 iterations are reported. As shown in Figure 26, the performance of our model using α and β with similar values is often better than other settings. In these real-world graph classification tasks, the constraints for *informativeness* and *representativeness* are equally important for our active feature and sample selection problem.

In 26(a), we find that the best parameter setting for NCI47 dataset is $\alpha = 0.001$, $\beta = 0.001$, and the accuracy is 63.5%. This setting is the same as our default parameter setting. The best parameter setting for NCI145 dataset is $\alpha = 0$, $\beta = 0.001$, and the resulting accuracy is 65.5%. Under our default setting, the accuracy is 65.4%. We find that the performance of our gActive model with the default setting is satisfactory. If we try to optimize the selection of α and β values, the accuracy improvement over other baselines will be even larger.

We also compare gActive models with and without pruning in the subgraph search space as summarized in Figure 27. The average CPU time with different *min_sup* during the first iteration is reported. gActive can improve the efficiencies by pruning the subgraph search space.

4.5 Conclusions

In this chapter we studied the dual active feature and sample selection problem for graph classification. The objective was to minimize the labeling efforts in graph classification. We demonstrated how to find a useful query graph and a set of optimal features simultaneously. We proposed to maximize the dependence between subgraph features and labels based upon an active learning framework. Our approach can find the most representative and informative graph and an optimal feature set. Then a branch-and-bound algorithm was proposed to prune the subgraph search space efficiently.

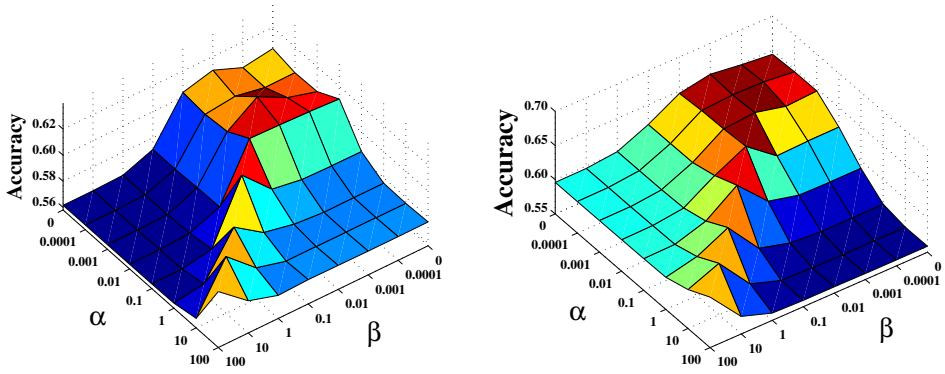


(a) NCI1 (b) NCI33 (c) NCI41

(d) NCI47 (e) NCI81 (f) NCI83

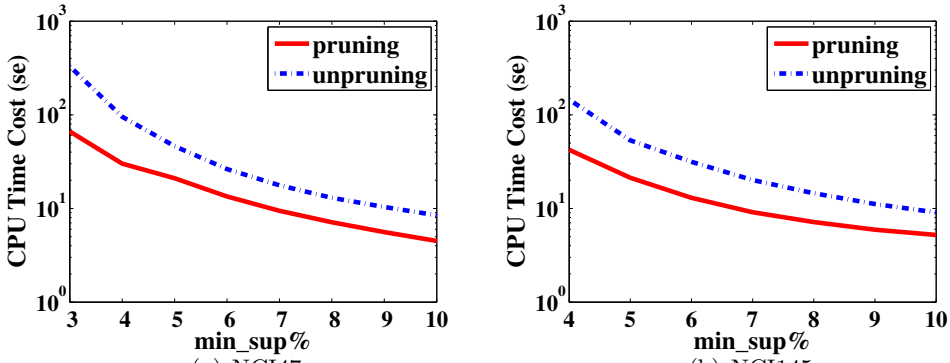
(g) NCI123 (h) NCI145 (i) AIDS

Figure 25. Graph classification accuracy after different number of graphs being queried.



(a) NCI47 (b) NCI145

Figure 26. gActive accuracies with different α and β.



(a) NCI47 (b) NCI145

Figure 27. CPU time with/without pruning.

CHAPTER 5

UNCERTAIN GRAPH CLASSIFICATION FOR BRAIN NETWORKS

5.1 Introduction

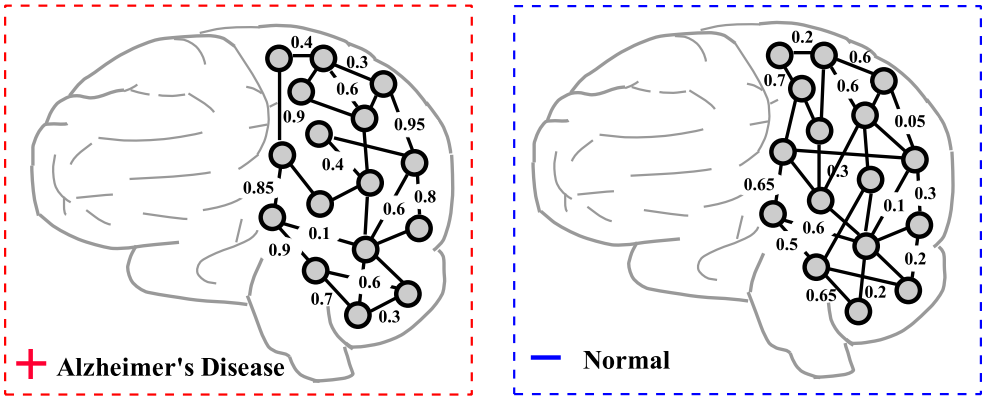
Much of the past research in discriminative subgraph feature mining has focused on certain graphs, where the structure of the graph objects are certain, and the binary edges represent the "presence" of linkages between the nodes. Conventional subgraph mining methods (13) utilize the structures of the certain graphs to find discriminative subgraph features. However, in many real-world applications, there is inherent uncertainty about the graph linkage structure. Such uncertainty information will be lost if we directly transform uncertain graphs into certain graphs.

For example, in neuroimaging, the functional connectivities among different brain regions are highly uncertain (5; 6; 7; 8). In such applications, each human brain can be represented as an uncertain graph as shown in Figure 28, which is also called the "brain network" (61). In such brain networks, the nodes represent brain regions, and edges represent the probabilistic connections, *e.g.*, resting-state functional connectivity in fMRI (functional Magnetic Resonance Imaging). Since these functional connectivities are derived based upon processing steps, such as temporal correlations in spontaneous blood oxygen level-dependent (BOLD) signal oscillations, each edge of the brain network is associated with a probability to quantify the likelihood that the functional connection exists in the brain. Resting-state functional connectivity has

shown alterations related to many neurological diseases, such as ADHD (Attention Deficit Hyperactivity Disorder), Alzheimer's disease and virus infections that may affect the brain functioning, such as HIV (62). Researchers are interested in analyzing the complex structure and uncertain connectivities of the human brain to find biomarkers for neurological diseases. Such biomarkers are clinically imperative for detecting injury to the brain in the earliest stages before it is irreversible. Valid biomarkers can be used to aid diagnosis, monitor disease progression and evaluate effects of intervention.

Motivated by these real-world neuroimaging applications, in this chapter, we study the problem of mining discriminative subgraph features in uncertain graph datasets. Discriminative subgraph features are fundamental for uncertain graphs, just as they are for certain graphs. They serve as primitive features for the classification tasks on uncertain graph objects. Despite the value and significance, the discriminative subgraph mining for uncertain graph classification has not been studied in this context. If we consider discriminative subgraph mining and uncertain graph structures as a whole, the major research challenges are as follows:

Structural Uncertainty: In discriminative subgraph mining, we need to estimate the discrimination score of a subgraph feature in order to select a set of subgraphs that are most discriminative for a classification task. In conventional subgraph mining, the discrimination scores of subgraph features are defined on certain graphs, where the structure of each graph object is certain, and thus the containment relationships between subgraph features and graph objects are also certain. However, when uncertainty is presented in the structures of graphs, a subgraph feature only exists within a graph object with a probability. Thus the discrimina-



(a) positive uncertain graph (b) negative uncertain graph

Figure 28. An example of uncertain graph classification task.

tion scores of a subgraph feature are no longer deterministic values, but random variables with probability distributions.

Thus, the evaluation of discrimination scores for subgraph features in uncertain graphs is different from conventional subgraph mining problems. For example, in Figure 29, we show an uncertain graph dataset containing 4 uncertain graphs $\widetilde{G}_1, \cdots, \widetilde{G}_4$ with their class labels, + or −. Subgraph g_1 is a frequent pattern among the uncertain graphs, but it may not relate to the class labels of the graphs. Subgraph g_2 is a discriminative subgraph features when we ignore the edge uncertainties. However, if such uncertainties are considered, we will find that g_2 can rarely be observed within the uncertain graph dataset, and thus will not be useful in graph classification. Accordingly, g_3 is the best subgraph feature for uncertain graph classification.

Efficiency & Robustness: There are two additional problems that need to be considered when evaluating features for uncertain graphs: 1) In an uncertain graph dataset, there are

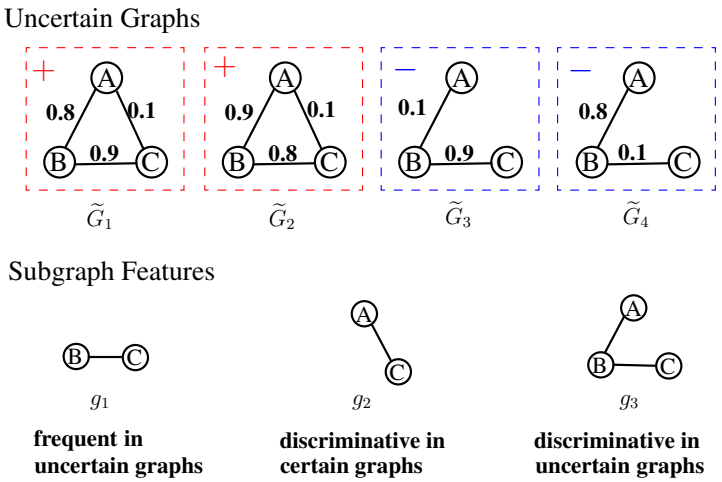


Figure 29. Different types of subgraph features for uncertain graph classification

an exponentially large number of possible instantiations of a graph dataset (16). How can we efficiently compute the discrimination score of a subgraph feature without enumerating all possible implied datasets? 2) When evaluating the subgraph features, we should choose a statistical measure for the probablity disctribution of discrimination scores which is robust to extreme values. For example, given a subgraph feature with (score, probability) pairs as $(0.01, 99.99\%)$ and $(+\infty, 0.01\%)$, the *expected* score of the subgraph is $+\infty$, although this value is only associated with a very tiny probability.

In order to address the above problems, we propose a general framework for mining discriminative subgraph features in uncertain graph datasets, which is called DUG (Discriminative feature selection for Uncertain Graph classification). The DUG framework can effectively find a set of discriminative subgraph features by considering the relationship between uncertain graph structures and labels based upon various statistical measures. We propose an efficient

TABLE VIII

IMPORTANT NOTATIONS.

Symbol	Definition
$\widetilde{\mathcal{D}} = \{\widetilde{G}_1, \cdots, \widetilde{G}_n\}$	uncertain graph dataset, $\widetilde{G}_i$ denotes the i-th uncertain graph in the dataset.
$\mathbf{y} = [y_1, \cdots, y_n]^\top$	class label vector for graphs in $\widetilde{\mathcal{D}}$, $y_i \in \{+1, -1\}$.
$\widetilde{\mathcal{D}}_+$ and $\widetilde{\mathcal{D}}_-$	the subset of $\widetilde{\mathcal{D}}$ with only positive/negative graphs, $\widetilde{\mathcal{D}}_+ = \{\widetilde{G}_i \| \widetilde{G}_i \in \widetilde{\mathcal{D}}, y_i = +1\}$.
n_+ and n_-	number of positive graphs and number of negative graphs in $\widetilde{\mathcal{D}}$, $n_+ = \|\widetilde{\mathcal{D}}_+\|$ and $n_- = \|\widetilde{\mathcal{D}}_-\|$.
$\mathcal{D} = \{G_1, \cdots, G_n\}$	a certain graph dataset implied from $\widetilde{\mathcal{D}}$, G_i denotes the certain graph implied from $\widetilde{G}_i$.
$g \subseteq G$	graph G contains subgraph feature g
n_+^g and n_-^g	number of graphs in $\mathcal{D}_+$ / $\mathcal{D}_-$ that contains subgraph g, $n_+^g = \|\{G_i \| g \subseteq G_i, G_i \in \mathcal{D}_+\}\|$.
$\widetilde{G} \Rightarrow G$ and $\widetilde{\mathcal{D}} \Rightarrow \mathcal{D}$	certain graph G is implied from uncertain graph $\widetilde{G}$; $\mathcal{D}$ is implied from $\widetilde{\mathcal{D}}$.
$\mathcal{W}(\widetilde{G})$ and $\mathcal{W}(\widetilde{\mathcal{D}})$	the possible worlds of $\widetilde{G}$ and $\widetilde{\mathcal{D}}$, $\mathcal{W}(\widetilde{G}) = \{G \| \widetilde{G} \Rightarrow G\}$, $\mathcal{W}(\widetilde{\mathcal{D}}) = \{\mathcal{D} \| \widetilde{\mathcal{D}} \Rightarrow \mathcal{D}\}$.
$E(\widetilde{G}_i)$ and $E(G_i)$	the set of edges in $\widetilde{G}_i$ and G_i
$\widetilde{\mathcal{D}}_+(k)$ and $\widetilde{\mathcal{D}}_-(k)$	the first k graphs in $\widetilde{\mathcal{D}}_+$ or $\widetilde{\mathcal{D}}_-$

method to calculate the probability distribution of the scoring function based on dynamic programming. Then a branch-and-bound algorithm is proposed to search for the discriminative subgraphs efficiently by pruning the subgraph search space. Empirical studies on resting-state fMRI images of different brain diseases (i.e., Alzheimer's Disease, ADHD and HIV) demonstrate that the proposed method can obtain better accuracy on uncertain graph classification tasks than alternative approaches.

For the rest of the chapter, we first introduce preliminaries in Section 5.3. Then we introduce our Dug subgraph mining framework in Section 5.4. Discrimination score functions based upon different statistic measures are discussed in Section 5.4.1. An efficient algorithm for computing the score distribution based upon dynamic programming is proposed in Section 5.4.2. Experimental results are discussed in Section 5.5. In Section 5.6, we conclude the chapter.

5.2 Related Work

Our work is related to subgraph mining techniques for uncertain graphs. Recently, there has been a growing interest in exploiting data uncertainty, especially structural uncertainty in graph data. There are some recent works on mining frequent subgraph features for uncertain graphs (63; 16; 64; 65). The problem of mining frequent subgraph in uncertain graphs are more difficult to those of certain graphs. The authors (63) proposed a method to estimate approximately the expected support of a subgraph feature in uncertain graph datasets. In (16), the authors studies the φ-probabilities for frequent subgraph features within uncertain graph datasets. The difference between these works and our chapter are as follows: 1) In this chapter, we study how to find discriminative subgraph features for uncertain graph data. The class labels of the graph objects are considered during the subgraph mining. 2) The graph model in our chapter is different from previous uncertain graph data, since we assume different graph object shares the same set of nodes as inspired by the neuroimaging applications. Thus, our method compute the exact discrimination scores of each subgraph features, instead of approximate scores. There are also many other works on uncertain graphs, which focus on different problems, *e.g.*, reliable subgraph mining (66), k-nearest neighbor discovery (67), subgraph retrieval (68) *etc.*

Our work is also motivated by the recent advances in analyzing neuroimaging data using data mining and machine learning approaches (5; 6; 7; 8). Huang et. al. (5) developed a sparse inverse covariance estimation method for analyzing brain connectivities in PET images of patients with Alzheimer's disease.

5.3 Problem Formulation

In this section, we formally define the model of uncertain graphs and the problem of discriminative subgraph mining in uncertain graph datasets. Suppose we are given an uncertain graph dataset $\widetilde{\mathcal{D}} = \{\widetilde{G}_1, \cdots, \widetilde{G}_n\}$ that consists of n uncertain graphs. $\widetilde{G}_i$ is the i-th uncertain graph in $\widetilde{\mathcal{D}}$. $\mathbf{y} = [y_1, \cdots, y_n]^\top$ denotes the vector of class labels, where $y_i \in \{+1, -1\}$ is the class label of $\widetilde{G}_i$. We also denote the subset of $\widetilde{\mathcal{D}}$ that contains only positive/negative graphs as $\widetilde{\mathcal{D}}_+ = \{\widetilde{G}_i | \widetilde{G}_i \in \widetilde{\mathcal{D}} \bigwedge y_i = +1\}$ and $\widetilde{\mathcal{D}}_- = \{\widetilde{G}_i | \widetilde{G}_i \in \widetilde{\mathcal{D}} \bigwedge y_i = -1\}$ respectively.

Definition 11 (Certain Graph) *A certain graph is an undirected and deterministic graph represented as $G = (V, E)$. $V = \{v_1, \cdots, v_{n_v}\}$ is the set of vertices. $E \subseteq V \times V$ is the set of deterministic edges.*

Definition 12 (Uncertain Graph) *An uncertain graph is an undirected and nondeterministic graph represented as $\widetilde{G} = (V, E, p)$. $V = \{v_1, \cdots, v_{n_v}\}$ is the set of vertices. $E \subseteq V \times V$ is the set of nondeterministic edges. $p : E \rightarrow (0, 1]$ is a function that assigns a probability of existence to each edge in E. $p(e)$ denotes the existence probability of edge $e \in E$.*

Consider an uncertain graph $\widetilde{G}(V, E, p) \in \widetilde{\mathcal{D}}$, where each edge $e \in E$ is associated with a probability $p(e)$ of being present. As in previous works (63; 16), we assume that the uncertainty variables of different edges in an uncertain graph are independent from each other, though most of our results are still applicable to graphs with edge correlations. We further assume that all uncertain graphs in a dataset $\widetilde{\mathcal{D}}$ share a same set of nodes V and each node in V has a unique node label, which is reasonable in many applications like neuroimaging, since each human brain

consists of the same number of regions. The main difference between different uncertain graphs is on their linkage structures, *i.e.*, the edge sets $E(\widetilde{G})$ and the edge probabilities $p(e)$.

Possible instantiations of an uncertain graph $\widetilde{G}$ are usually referred to as *worlds* of $\widetilde{G}$, where each world corresponds to an implied certain graph G. Here G is *implied* from uncertain graph $\widetilde{G}$ (denoted as $\widetilde{G} \Rightarrow G$), iff all edges in $E(G)$ are sampled from $E(\widetilde{G})$ according to their probabilities of existence in $p(e)$ and $E(G) \subseteq E(\widetilde{G})$. There are $2^{|E(\widetilde{G})|}$ possible worlds for uncertain graph $\widetilde{G}$, denoted as $\mathcal{W}(\widetilde{G}) = \{G | \widetilde{G} \Rightarrow G\}$. Thus, each uncertain graph $\widetilde{G}$ corresponds to a probability distribution over $\mathcal{W}(\widetilde{G})$. We denote the probability of each certain graph $G \in \mathcal{W}(\widetilde{G})$ being implied by the uncertain graph $\widetilde{G}$ as $\Pr(\widetilde{G} \Rightarrow G)$, and we have

$$\Pr\Big[\widetilde{G} \Rightarrow G\Big] = \prod_{e \in E(G)} \Pr_{\widetilde{G}}(e) \prod_{e \in E(\widetilde{G}) - E(G)} \big(1 - \Pr_{\widetilde{G}}(e) \big)$$

Similarly, possible instantiations of an uncertain graph dataset $\widetilde{\mathcal{D}} = \{\widetilde{G}_1, \cdots, \widetilde{G}_n\}$ are referred to as *worlds* of $\widetilde{\mathcal{D}}$, where each world corresponds to an implied certain graph dataset $\mathcal{D} = \{G_1, \cdots, G_n\}$. A certain graph dataset $\mathcal{D}$ is called as being *implied* from uncertain graph dataset $\widetilde{\mathcal{D}}$ (denoted as $\widetilde{\mathcal{D}} \Rightarrow \mathcal{D}$), iff $|\mathcal{D}| = |\widetilde{\mathcal{D}}|$ and $\forall i \in \{1, \cdots, |\mathcal{D}|\}$, $\widetilde{G}_i \Rightarrow G_i$. There are $\prod_{i=1}^{|\widetilde{\mathcal{D}}|} 2^{|E(\widetilde{G}_i)|}$ possible worlds for uncertain graph dataset $\widetilde{\mathcal{D}}$, denoted as $\mathcal{W}(\widetilde{\mathcal{D}}) = \{\mathcal{D} \mid \widetilde{\mathcal{D}} \Rightarrow \mathcal{D}\}$. An uncertain graph dataset $\widetilde{\mathcal{D}}$ corresponds to a probability distribution over $\mathcal{W}(\widetilde{\mathcal{D}})$. We denote

the probability of each certain graph dataset $\mathcal{D} \in \mathcal{W}(\widetilde{\mathcal{D}})$ being implied by $\widetilde{\mathcal{D}}$ as $\Pr(\widetilde{\mathcal{D}} \Rightarrow \mathcal{D})$. By assuming that different uncertain graphs are independent from each other, we have

$$\Pr\left[\widetilde{\mathcal{D}} \Rightarrow \mathcal{D}\right] = \prod_{i=1}^{|\widetilde{\mathcal{D}}|} \Pr[\widetilde{G}_i \Rightarrow G_i]$$

The concept of *subgraph* is defined based upon certain graphs. Different from conventional subgraph mining problems where each subgraph feature can have multiple embeddings within one graph object, in our data model, each subgraph feature g can only have one unique embedding within a certain graph G.

Definition 13 (Subgraph) *Let $g = (V', E')$ and $G = (V, E)$ be two certain graphs. g is a subgraph of G (denoted as $g \subseteq G$) iff $V' \subseteq V$ and $E' \subseteq E$. We use $g \subseteq G$ to denote that graph g is a subgraph of G. We also say that G contains subgraph g.*

For an uncertain graph $\widetilde{G}$, the probability of $\widetilde{G}$ containing a subgraph feature g is defined as follows:

$$\begin{aligned}\Pr(g \subseteq \widetilde{G}) &= \sum_{G \in \mathcal{W}(\widetilde{G})} \Pr(\widetilde{G} \Rightarrow G) \cdot I(g \subseteq G) \\ &= \begin{cases} \prod_{e \in E(g)} p(e) & \text{if } E(g) \subseteq E(\widetilde{G}) \\ 0 & \text{otherwise} \end{cases}\end{aligned}$$

which corresponds to the probability that a certain graph G implied by $\widetilde{G}$ contains subgraph g.

We focus on mining a set of discriminative subgraph features to define the feature space of graph classification. It is assumed that a graph object $\widetilde{G}_i$ is represented as a feature vector $\mathbf{x}_i = [x_i^1, \cdots, x_i^m]^\top$ associated with a set of subgraph features $\{g_1, \cdots, g_m\}$. Here, $x_i^k = \Pr(g_k \subseteq \widetilde{G}_i)$ is the probability that $\widetilde{G}_i$ contains the subgraph feature g_k. Now suppose the full set of subgraph features in the graph dataset $\widetilde{\mathcal{D}}$ is $\mathcal{S} = \{g_1, \cdots, g_m\}$, which we use to predict the class labels of the graph objects. The full feature set $\mathcal{S}$ is very large. Only a subset of the subgraph features ($\mathcal{T} \subseteq \mathcal{S}$) is relevant to the graph classification task, which is the target feature set we want to find within uncertain graphs.

The key issues of discriminative subgraph mining for uncertain graphs can be described as follows:

(P1) How can one properly evaluate the discrimination scores of a subgraph feature considering the uncertainty of the graph structures?

(P2) How can one efficiently compute the probability distribution of a subgraph's discrimination score by avoiding the exhaustive enumeration of all possible worlds of the uncertain graph dataset? Moreover, since the subgraph enumeration is NP-hard, it is also infeasible to fully enumerate all the subgraph features for an uncertain graph dataset.

In the following sections, we will introduce the proposed framework for mining discriminative subgraphs from uncertain graphs.

5.4 The Proposed Framework

5.4.1 Discrimination Score Distribution

In this subsection, we address the problem (P1) discussed in the previous section. In conventional discriminative subgraph mining, the discrimination scores of subgraph features are usually defined for certain graph datasets, e.g., information gain and G-test score (13). The score of a subgraph feature is a fixed value indicating the discriminative power of the subgraph feature for the graph classification task. However, such concepts don't make sense to uncertain graph datasets, since an uncertain graph only contains a subgraph feature in a probabilistic sense. Now we extend the concept of discriminative subgraph features in uncertain graph datasets. Suppose we have an objective function $F(g, \mathcal{D})$ which measures the discrimination score of a subgraph g in a certain graph dataset $\mathcal{D}$. The corresponding objective function on an uncertain graph dataset $\widetilde{\mathcal{D}}$ can be written as $F(g, \widetilde{\mathcal{D}})$ accordingly. Note that $F(g, \widetilde{\mathcal{D}})$ is no longer a deterministic function. $F(g, \widetilde{\mathcal{D}})$ corresponds to a random variable over all possible outcomes of $F(g, \mathcal{D})$ (*i.e.*, Range(F)) with probability distribution:

s_1	s_2	$\cdots$	s_i	$\cdots$
$\Pr[F(g, \widetilde{\mathcal{D}}) = s_1]$	$\Pr[F(g, \widetilde{\mathcal{D}}) = s_2]$	$\cdots$	$\Pr[F(g, \widetilde{\mathcal{D}}) = s_i]$	$\cdots$

where $s_i \in \text{Range}(F)$.

The probability distribution of the discrimination score values can be defined as follows:

$$\Pr\Big[F(g, \widetilde{\mathcal{D}}) = s\Big] = \sum_{\mathcal{D} \in \mathcal{W}(\widetilde{\mathcal{D}})} \Pr[\widetilde{\mathcal{D}} \Rightarrow D] \cdot I\left(F(g, \mathcal{D}) = s\right)$$

where $I(\pi) \in \{0, 1\}$ is an indicator function, and $I(\pi) = 1$ iff π holds. In other words, $\forall s \in \mathit{Range}(F)$, $\Pr[F(g, \widetilde{\mathcal{D}}) = s]$ is the summation over the probabilities of all worlds of $\widetilde{\mathcal{D}}$ in which the discrimination score $F(g, \mathcal{D})$ is exactly s. Based on the discrimination score function on uncertain graphs, we define four statistical measures that evaluate the properties of the distribution of $F(g, \widetilde{\mathcal{D}})$ from different perspectives.

Definition 14 (Mean-Score) *Given an uncertain graph dataset $\widetilde{\mathcal{D}}$, a subgraph feature g and a discrimination score function $F(\cdot, \cdot)$, we define the expected discrimination score* $\textsc{Exp}(F(g, \widetilde{\mathcal{D}}))$ *as the mean score among all possible worlds of $\widetilde{\mathcal{D}}$:*

$$\begin{aligned}\textsc{Exp}\Big(F(g, \widetilde{\mathcal{D}})\Big) &= \sum_{\mathcal{D} \in \mathcal{W}(\widetilde{\mathcal{D}})} \Pr[\widetilde{\mathcal{D}} \Rightarrow D] \cdot F(g, \mathcal{D}) \\ &= \sum_{s=-\infty}^{+\infty} s \cdot \Pr[F(g, \widetilde{\mathcal{D}}) = s]\end{aligned}$$

The mean discrimination score is the expectation of the random variable $F(g, \widetilde{\mathcal{D}})$. The expectation is usually used in conventional frequent pattern mining on uncertain datasets (63; 16). However, it's worth noting that the expectation of discrimination scores may not be robust to extreme values. In discriminative subgraph mining, the value of a score function (*e.g.*, frequency ratio(69), G-test score(13)) can be $+\infty$. Such cases can easily dominate the computation of expectation, even if the probabilities are extremely small. For example, suppose we have a subgraph feature with the (score, probability) pairs as $(0.01, 99.99\%)$ and $(+\infty, 0.01\%)$. The expected score will be $+\infty$. In order to address this problem, we either need to bound the

maximum value of the objective function like $\min(F(g, \widetilde{\mathcal{D}}), \frac{1}{\epsilon})$, or we need to introduce other statistical measures which are robust to extreme values.

Definition 15 (Median-Score) *Given an uncertain graph dataset $\widetilde{\mathcal{D}}$, a subgraph feature g and a discrimination score function $F(\cdot, \cdot)$ on certain graphs, we define the median discrimination score* $\textsc{Median}(F(g, \widetilde{\mathcal{D}}))$ *as the median score among all possible worlds of $\widetilde{\mathcal{D}}$:*

$$\textsc{Median}\Big(F(g, \widetilde{\mathcal{D}})\Big) = \arg\max_{S} \left\{ \sum_{s=-\infty}^{S} \Pr\Big[F(g, \widetilde{\mathcal{D}}) = s\Big] \ \leq \frac{1}{2} \right\}$$

The median score is relatively more robust to extreme values than expectation, although in some cases the median score can still be infinite. The same results can also hold for any quantile or k-th order statistic.

Another commonly used statistic is the mode score, *i.e.*, the score value that has the largest probability. The mode score of a distribution means that the score is most likely to be observed within all possible worlds of $\widetilde{\mathcal{D}}$.

Definition 16 (Mode-Score) *Given an uncertain graph dataset $\widetilde{\mathcal{D}}$, a subgraph feature g and a discrimination score function $F(\cdot, \cdot)$, we define the mode discrimination score* $\textsc{Mode}(F(g, \widetilde{\mathcal{D}}))$ *as the score that is most likely among all possible worlds of $\widetilde{\mathcal{D}}$:*

$$\textsc{Mode}\Big(F(g, \widetilde{\mathcal{D}})\Big) = \arg\max_{s} \Pr\Big[F(g, \widetilde{\mathcal{D}}) = s\Big]$$

Next we consider the probability of a subgraph feature being observed as a discriminative pattern within all possible worlds of $\widetilde{\mathcal{D}}$, *i.e.*, $\Pr[F(g, \widetilde{\mathcal{D}}) \geq \varphi]$. It is called φ-probability. The higher the value, the more likely that the subgraph feature is a discriminative pattern with a score larger or equals to a threshold φ.

Definition 17 (φ-Probability) *Given an uncertain graph dataset $\widetilde{\mathcal{D}}$, a subgraph feature g and a discrimination score function $F(\cdot, \cdot)$, we define the φ-probability for discrimination score function $F(g, \widetilde{\mathcal{D}})$ as the sum of probabilities for all possible worlds of $\widetilde{\mathcal{D}}$, where the score is greater than or equals to φ:*

$$\begin{aligned} \varphi\text{-}\Pr\Big(F(g, \widetilde{\mathcal{D}})\Big) &= \sum_{\mathcal{D} \in \mathcal{W}(\widetilde{\mathcal{D}})} \Pr[\widetilde{\mathcal{D}} \Rightarrow D] \cdot I\left(F(g, \mathcal{D}) \geq \varphi\right) \\ &= \sum_{s=\varphi}^{+\infty} \Pr[F(g, \widetilde{\mathcal{D}}) = s] \end{aligned}$$

The φ-probability is robust to extreme values of the objective function. For the previous example, we have a subgraph feature with score distribution: $(0.01, 99.99\%), (+\infty, 0.01\%)$. The φ-probability is 0.01%, when $\varphi = 1$.

We have already introduced four statistical measures of the distribution of a discrimination score function. Now the central problem for calculating all these measures is how to calculate $\Pr[F(g, \widetilde{\mathcal{D}}) = s]$ efficiently, which we will discuss in the following section.

5.4.2 Efficient Computation

In this subsection, we address the problem (P2) discussed in Section 5.3. Given a certain graph dataset $\mathcal{D}$, we denote the subsets of all positive graphs and all negative graphs as $\mathcal{D}_+$

TABLE IX

SUMMARY OF DISCRIMINATION SCORE FUNCTIONS.

Name	$f(n_+^g, n_-^g, n_+, n_-)$
confidence	$\frac{n_+^g}{n_+^g+n_-^g}$
frequency ratio	$\left\lvert\log \frac{n_+^g \cdot n_-}{n_-^g \cdot n_+}\right\rvert$
G-test	$2n_+^g \cdot \ln\frac{n_+^g \cdot n_-}{n_-^g \cdot n_+} + 2(n_+ - n_+^g) \cdot \ln\frac{n_- \cdot (n_+ - n_+^g)}{n_+ \cdot (n_- - n_-^g)}$
HSIC(linear)	$\frac{(n_+^g \cdot n_- - n_-^g \cdot n_+)^2}{(n_+ + n_- - 1)^2 (n_+ + n_-)^2}$

and $\mathcal{D}_-$, respectively. Suppose the supports of subgraph feature g in $\mathcal{D}_+$ and $\mathcal{D}_-$ are n_+^g and n_-^g. $n_+^g = |\{G; G \in \mathcal{D}_+, g \subseteq G\}|$. Most of the existing discrimination score functions can be written as a function of n_+^g, n_-^g, n_+ and n_-:

$$F(g, \mathcal{D}) = f\big(n_+^g, n_-^g, n_+, n_-\big) \tag{5.1}$$

The definition in Equation 5.1 covers many discrimination score functions including confidence(70), frequency ratio(69), information gain, G-test score(13) and HSIC(71), as shown in Table IX. For example, frequency ratio can be written as $\mathrm{r}(g) = |\log \frac{n_+^g \cdot n_-}{n_-^g \cdot n_+}|$. The G-test score can be written as $\text{G-test}(g) = 2n_+^g \cdot \ln \frac{n_+^g \cdot n_-}{n_-^g \cdot n_+} + 2\left(n_+ - n_+^g\right) \cdot \ln \frac{n_- \cdot (n_+ - n_+^g)}{n_+ \cdot (n_- - n_-^g)}$. Because n_+ and n_- are fixed numbers for different subgraph features, we simply use $f(n_+^g, n_-^g)$ for $f\big(n_+^g, n_-^g, n_+, n_-\big)$.

Based on the above definitions, we find that the number of possible outcomes of $F(g, \widetilde{\mathcal{D}})$ is bounded by $n_+ \times n_-$, because $0 \leq n_+^g \leq n_+$ and $0 \leq n_-^g \leq n_-$. Thus, the probabilities

$\Pr[F(g, \widetilde{\mathcal{D}}) = s]$ can be exactly computed via dynamic programming in $O(n^2)$ time, without enumerating all possible worlds of $\widetilde{\mathcal{D}}$. Instead, we can just enumerate all possible combinations of (n_+^g, n_-^g) and calculate the probability for each pair (n_+^g, n_-^g), denoted as $\Pr[n_+^g, n_-^g, \widetilde{\mathcal{D}}] = \Pr\Big[F(g, \widetilde{\mathcal{D}}) = f(n_+^g, n_-^g)\Big]$. Then the values of $F(g, \widetilde{\mathcal{D}})$ in all possible worlds with non-zero probabilities can be covered by the $n_+ \times n_-$ cases.

Moreover, because different uncertain graphs are independent from each other, we have

$$\Pr[n_+^g, n_-^g, \widetilde{\mathcal{D}}] = \Pr[n_+^g, \widetilde{\mathcal{D}}_+] \cdot \Pr[n_-^g, \widetilde{\mathcal{D}}_-] \tag{5.2}$$

where $\Pr[n_+^g, \widetilde{\mathcal{D}}_+]$ denotes the probability of the cases when there are n_+^g graphs in $\widetilde{\mathcal{D}}_+$ that contain the subgraph g. $\Pr[n_-^g, \widetilde{\mathcal{D}}_-]$ corresponds to the cases when there are n_-^g graphs in $\widetilde{\mathcal{D}}_-$ that contain subgraph g. Now we just need to compute the probabilities $\Pr[n_+^g, \widetilde{\mathcal{D}}_+]$ $(\forall n_+^g, 0 \leq n_+^g \leq n_+)$ and $\Pr[n_-^g, \widetilde{\mathcal{D}}_-]$ $(\forall n_-^g, 0 \leq n_-^g \leq n_-)$ separately.

Let $\widetilde{\mathcal{D}}(k)$ denote the first k uncertain graphs in $\widetilde{\mathcal{D}}$, *i.e.*, $\widetilde{\mathcal{D}}(k) = \{\widetilde{G}_1, \cdots, \widetilde{G}_k\}$. $\widetilde{\mathcal{D}}_+(k)$ and $\widetilde{\mathcal{D}}_-(k)$ denote the first k graphs in $\widetilde{\mathcal{D}}_+$ and $\widetilde{\mathcal{D}}_-$ respectively. All the values of $\Pr[n_+^g, \widetilde{\mathcal{D}}_+]$ and $\Pr[n_-^g, \widetilde{\mathcal{D}}_-]$ can be calculated using the recursive equation in Figure 32. The $\Pr[i, \widetilde{\mathcal{D}}(k)]$ denotes the probability when there are i graphs containing g in $\widetilde{\mathcal{D}}(k)$. And the target values to calculate are $\Pr[i, \widetilde{\mathcal{D}}_+(n_+)]$ $(\forall i, 0 \leq i \leq n_+)$ and $\Pr[i, \widetilde{\mathcal{D}}_-(n_-)]$ $(\forall i, 0 \leq i \leq n_-)$ by substituting the $\widetilde{\mathcal{D}}_+$ and $\widetilde{\mathcal{D}}_-$ into the Equation 5.3, respectively. In Figure 31, we showed the dynamic programing algorithm to compute the target values using Equation 5.3. Figure 30 illustrates

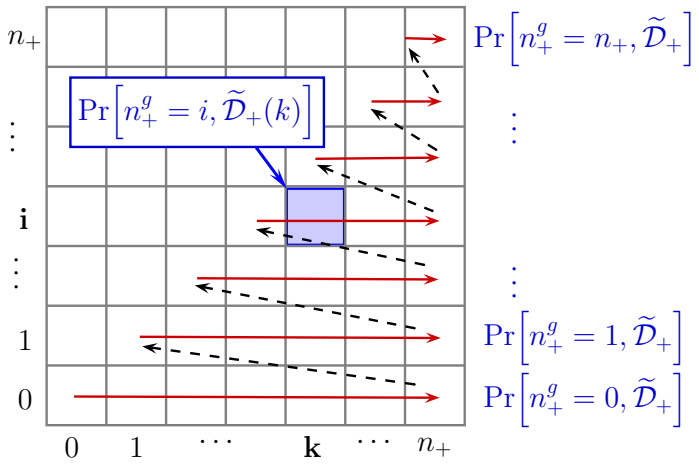


Figure 30. The dynamic programming process for computing $\Pr\left[n_+^g, \widetilde{\mathcal{D}}_+\right]$.

the computation process of the dynamic programing algorithm for $\Pr[n_+^g, \widetilde{\mathcal{D}}_+]$, while the same process also applies for $\Pr[n_-^g, \widetilde{\mathcal{D}}_-]$.

For details of the recursive equations in Figure 32, we have the base cases, $\Pr[0, \widetilde{\mathcal{D}}_0] = 1$ and $\Pr[i, \widetilde{\mathcal{D}}(k)] = 0$ (if $i > k$ or $i < 0$). For other cases, the probability value can be calculated through the recursive equation in Equation 5.3. Then, $\Pr[n_+^g, n_-^g, \widetilde{\mathcal{D}}]$ can be calculated via Equation 5.2. Thus all the statistical measures mentioned in Section 5.4.1 can be calculated within $O(n^2)$ time as follows:

$$\textsc{Exp}\left(F(g, \widetilde{\mathcal{D}})\right) = \sum_{n_+^g=0}^{n_+} \sum_{n_-^g=0}^{n_-} \Pr[n_+^g, n_-^g, \widetilde{\mathcal{D}})] \cdot f(n_+^g, n_-^g)$$

Input:
$\widetilde{\mathcal{D}}_+$: the set of positive graphs
$\widetilde{\mathcal{D}}_-$: the set of negative graphs

Dynamic Programming:
for $n_+^g \leftarrow 0$ to n_+
  for $k \leftarrow n_+^g$ to n_+
    compute Pr[$n_+^g, \widetilde{\mathcal{D}}_+(k)$] via Equation 5.3;
for $n_-^g \leftarrow 0$ to n_-
  for $k \leftarrow n_-^g$ to n_-
    compute Pr[$n_-^g, \widetilde{\mathcal{D}}_-(k)$] via Equation 5.3;

Output:
$\Pr[n_+^g, \widetilde{\mathcal{D}}_+]$ $(\forall\ n_+^g,\ 0 \leq n_+^g \leq n_+)$
$\Pr[n_-^g, \widetilde{\mathcal{D}}_-]$ $(\forall\ n_-^g,\ 0 \leq n_-^g \leq n_-)$

Figure 31. The dynamic programming algorithm for probability computation.

$$\textsc{Median}\Big(F(g, \widetilde{\mathcal{D}})\Big) = \arg\max_s \left\{ \sum_{x=-\infty}^{s} \sum_{f(n_+^g, n_-^g)=x} Pr[n_+^g, n_-^g, \widetilde{\mathcal{D}})] \ \ \leq \frac{1}{2} \right\}$$

$$\textsc{Mode}\Big(F(g, \widetilde{\mathcal{D}})\Big) = \arg\max_s \sum_{n_+^g=0}^{n_+} \sum_{n_-^g=0}^{n_-} \Pr[n_+^g, n_-^g, \widetilde{\mathcal{D}})] \cdot I(f(n_+^g, n_-^g) = s)$$

$$\varphi\text{-}\Pr\Big(F(g, \widetilde{\mathcal{D}})\Big) = \sum_{n_+^g=0}^{n_+} \sum_{n_-^g=0}^{n_-} \Pr[n_+^g, n_-^g, \widetilde{\mathcal{D}})] \cdot I(f(n_+^g, n_-^g) \geq \varphi)$$

We will show later that the dynamic programming process is highly efficient in all the applications studied in Section 5.5. For dataset with even larger number of graphs, the divid-and-conquer method in (72) could also be used here to further optimize the computational cost.

$$\Pr\Big[i, \widetilde{\mathcal{D}}(k)\Big] = \begin{cases} \Big(1 - \Pr[g \subseteq \widetilde{G}_k]\Big) \cdot \Pr[i, \widetilde{\mathcal{D}}(k-1)] \ + \ \Pr[g \subseteq \widetilde{G}(k)] \cdot \Pr[i-1, \widetilde{\mathcal{D}}(k-1)] & \text{if } i \leq k \\ 1 & \text{if } i = k = 0 \\ 0 & \text{if } i > k \text{ or } i < 0 \end{cases} \tag{5.3}$$

Figure 32. Recursive equation for dynamic programming.

Input:

$\widetilde{\mathcal{D}}$: the uncertain graph dataset $\{\widetilde{G}_1, \cdots, \widetilde{G}_n\}$
t: the maximum number of subgraphs.
$\mathbf{y}$: the vector of class labels for uncertain graphs,
min_sup: the minimum expected frequency.
M: the statistic measure (Expectation/Median/Mode/φ-Pr)

Recursive Subgraphs Mining:

- Depth-first search the gSpan's code tree and update the feature list as follows:
  1. Update the candidate feature list using the current subgraph feature g_c:
     Calculate the probability vector $\Pr[n_+^{g_c}, \widetilde{\mathcal{D}}_+]$ and $\Pr[n_-^{g_c}, \widetilde{\mathcal{D}}_-]$ using the dynamic programing algorithm in Figure 31
     Compute the statistic measure $M\left(F(g_c, \widetilde{\mathcal{D}})\right)$ based on the discrimination score function $F(g_c, \widetilde{\mathcal{D}})$.
     If the score is larger than the worst feature in $\mathcal{T}$, replace it and update $\theta = \min_{g \in \mathcal{T}} M\left(F(g, \widetilde{\mathcal{D}})\right)$
  2. Test pruning criteria for the sub-tree rooted from node g as follows:
     if Exp-Freq$(g_c) \leq$ *min_sup*, prune the sub-tree of g_c
     if Bound-$M\left(F(g_c, \widetilde{\mathcal{D}})\right) \leq \theta$, prune the sub-tree of g_c
  3. Recursion: Depth-first search the sub-tree rooted from node g_c

Output:

$\mathcal{T}$: the discriminative subgraph features for uncertain graph classification.

Figure 33. The DUG framework for discriminative subgraph mining.

5.4.3 Upper-Bounds for Subgraph Pruning

In order to avoid the exhaustive enumeration of subgraph features, we derive some subgraph pruning methods. One natural pruning bound for subgraph search is the expected frequency of a subgraph feature, Exp-Freq$(g, \widetilde{\mathcal{D}}) = \frac{\sum_i^n \Pr(g \subseteq \widetilde{\mathcal{G}}_i)}{n}$, since it's can be easily proved with anti-monotonic property. For the expectation and φ-probability, we can also derive additional bounds for subgraph pruning. Let $\hat{F}(g, \mathcal{D}) = \hat{f}(n_+^g, n_-^g)$ be the estimated upper-bound function

for g and its supergraphs in certain graph dataset $\mathcal{D}$. We can derive the corresponding upper-bounds as follows:

$$\text{UB-Exp}(g, \widetilde{\mathcal{D}}) = \sum_{n_+^g=0}^{n_+} \sum_{n_-^g=0}^{n_-} \Pr[n_+^g, n_-^g, \widetilde{\mathcal{D}})] \cdot \hat{f}(n_+^g, n_-^g)$$

$$\text{UB-}\varphi\text{-Pr}(g, \widetilde{\mathcal{D}}) = \sum_{n_+^g=0}^{n_+} \sum_{n_-^g=0}^{n_-} \Pr[n_+^g, n_-^g, \widetilde{\mathcal{D}})] \cdot I(\hat{f}(n_+^g, n_-^g) \geq \varphi)$$

For the median and mode measures, it is difficult to derive a meaningful bound, thus we simply use the expected frequency to perform the subgraph pruning.

We now utilize the above bounds to prune the DFS-code tree in gSpan (28) by the branch-and-bound pruning. The top-t best features are maintained in a candidate list. During the subgraph mining, we calculate the upper-bound of each subgraph feature in the search tree. If a subgraph feature with its children pattern cannot update the candidate feature list, we can prune the subtree of gSpan rooted from this node. It is guaranteed by the upper-bounds that we will not miss any better subgraph features. Thus, the subgraph mining process can be speeded up without loss of performance. The algorithm of Dug is summarized in Figure 33.

5.5 Experiments

In order to evaluate the performance of the proposed approach for uncertain graph classification, we tested our algorithm on real-world fMRI brain images as summarized in Table X.

5.5.1 Data Collection

In order to evaluate the performance of the proposed approach for uncertain graph classification, we tested our algorithm on real-world fMRI brain images.

- *Alzheimer's Disease (ADNI)*: The first dataset is collected from the Alzheimer's Disease Neuroimaging Initiative[1]. The dataset consists of records of patients with Alzheimer's Disease (AD) and Mild Cognitive Impairment (MCI). We downloaded all records of resting-state fMRI images and treated the normal brains as *negative* graphs, and AD+MCI as the *positive* graphs. We applyed Automated Anatomical Labeling (AAL[2]) to extract a sequence of responds from each of of the 116 anatomical volumes of interest (AVOI), where each AVOI represents a different brain region. The correlations of brain activities among different brain regions are computed. Positive correlations are used as uncertain links among brain regions. For details, we used SPM8 toolbox[3], and functional images were realigned to the first volume, slice timing corrected, and normalized to the MNI template and spatially smoothed with an 8-mm Gaussian kernel. Resting-State fMRI Data Analysis Toolkit (REST[4]) was then used to remove the linear trend of time series and

---

[1]http://adni.loni.ucla.edu/

[2]`http://www.cyceron.fr/web/aal__anatomical_automatic_labeling.html`

[3]http://www.fil.ion.ucl.ac.uk/spm/software/spm8/

[4]http://resting-fmri.sourceforge.net

temporally band-pass filtering (0.01-0.08 Hz). Each brain is represented as an uncertain graph with 90 nodes corresponding to 90 cerebral regions, excluding 26 cerebellar regions.

- *Attention Deficit Hyperactivity Disorder (ADHD)*: The second dataset is collected from ADHD-200 global competition dataset [1]. The dataset contains records of resting-state fMRI images for 776 subjects, which are labeled as real patients (positive) and normal controls (negative). Similar to the ADNI dataset, the brain images are preprocessed using Athena Pipeline[2]. The original dataset is unbalanced, we randomly sampled 100 ADHD patients and 100 normal controls from the dataset for performance evaluation.
- *Human Immunodeficiency Virus Infection (HIV)*: The third dataset is collected from the Chicago Early HIV Infection Study in Northwestern University (62). The dataset contains fMRI brain images of patients with early HIV infection (positive) as well as normal controls (negative). The same preprocessing steps as in ADNI dataset were used to extract a functional connectivity network from each image.

5.5.2 Comparative Methods

We compared our method using different statistical measures and discrimination score functions summarized as follows:

- *Frequent Subgraphs + Expectation (Exp+Freq)*: The first baseline method is finding frequent subgraph features within uncertain graphs. This baseline is similar to the method

---

[1] http://neurobureau.projects.nitrc.org/ADHD200/

[2] `http://www.nitrc.org/plugins/mwiki/index.php/neurobureau:AthenaPipeline`

TABLE X

SUMMARY OF EXPERIMENTAL DATASETS.

	$\lvert\widetilde{\mathcal{D}}\rvert$	$\lvert\widetilde{\mathcal{D}}_+\rvert$	$\lvert\widetilde{\mathcal{D}}_-\rvert$	$\lvert V\rvert$	avg. $\lvert E\rvert$	avg. edge prob
ADHD	200	100	100	116	484.7	0.55
ADNI	36	18	18	90	2019.8	0.59
HIV	50	25	25	90	480.48	0.88

introduced in (63). In our data model, this baseline method computes the exact expected frequency of each subgraph features, instead of approximated values. The top ranked frequent patterns are extracted as used as features for graph classification.

- DUG *with HSIC based discrimination scores*: we compare with four different versions of our DUG method based upon HSIC criterion, which maximize the dependence between subgraph features and graph labels (71). "Exp-HSIC" computes the expected HSIC value for each subgraph feature, and find the top-k subgraphs with the largest values. "Med-HSIC" computes the median HSIC value for each subgraph feature, while "Mod-HSIC" computes the mode HSIC value. "φ Pr-HSIC" computes the φ-probability of HSIC value for each subgraph feature.

- DUG *with Frequency Ratio based discrimination scores*: we also compare our method based upon Frequency Ratio, *i.e.*, "Exp-Ratio", "Med-Ratio", "Mod-Ratio" and "φ Pr-Ratio".

- DUG *with G-test based discrimination scores*: we then compare our method based upon G-test criterion, *i.e.*, "Exp-Gtest", "Med-Gtest", "Mod-Gtest" and "φ Pr-Gtest".

TABLE XI. Results on the ADNI (Alzheimer's Disease) dataset with different number of features($t = 100, \cdots, 500$). The results are reported as "average performance + (rank)".

	Error Rate ↓					F1 ↑				
Methods	$t = 100$	$t = 200$	$t = 300$	$t = 400$	$t = 500$	$t = 100$	$t = 200$	$t = 300$	$t = 400$	$t = 500$
Exp-HSIC	0.400 (9)	0.367 (8)	0.367 (10)	0.317 (4)	0.333 (9)	0.699 (9)	0.725 (9)	0.725 (9)	0.753 (6)	0.743 (10)
Med-HSIC	0.433 (14)	0.350 (5)	0.333 (6)	0.350 (8)	0.317 (7)	0.667 (13)	0.741 (7)	0.757 (4)	0.734 (9)	0.766 (7)
Mod-HSIC	0.367 (6)	0.333 (3)	0.300 (1)*	0.317 (4)	0.300 (2)	0.703 (8)	0.750 (4)	0.776 (3)	0.766 (3)	0.775 (4)
φPr-HSIC	0.283 (1)*	0.283 (1)*	0.333 (6)	0.333 (7)	0.300 (2)	0.778 (1)*	0.785 (1)*	0.757 (4)	0.750 (7)	0.776 (3)
HSIC	0.450 (16)	0.467 (19)	0.467 (17)	0.500 (18)	0.500 (18)	0.615 (18)	0.597 (19)	0.622 (17)	0.583 (18)	0.584 (18)
Exp-Ratio	0.433 (14)	0.383 (10)	0.317 (4)	0.300 (2)	0.300 (2)	0.667 (13)	0.715 (10)	0.756 (6)	0.766 (3)	0.766 (7)
Med-Ratio	0.450 (16)	0.417 (15)	0.450 (16)	0.383 (11)	0.383 (11)	0.639 (17)	0.653 (16)	0.608 (20)	0.689 (12)	0.684 (11)
Mod-Ratio	0.317 (3)	0.350 (5)	0.433 (15)	0.417 (13)	0.467 (15)	0.776 (2)	0.744 (6)	0.659 (13)	0.657 (13)	0.612 (15)
φPr-Ratio	0.400 (9)	0.317 (2)	0.300 (1)*	0.300 (2)	0.267 (1)*	0.692 (10)	0.764 (2)	0.784 (1)*	0.778 (2)	0.809 (1)*
Ratio	0.500 (19)	0.483 (20)	0.533 (22)	0.567 (22)	0.533 (20)	0.581 (20)	0.603 (18)	0.533 (21)	0.519 (22)	0.550 (20)
Exp-Gtest	0.300 (2)	0.367 (8)	0.317 (4)	0.350 (8)	0.383 (11)	0.774 (3)	0.693 (11)	0.729 (9)	0.702 (10)	0.672 (12)
Med-Gtest	0.517 (21)	0.450 (18)	0.400 (11)	0.500 (18)	0.483 (17)	0.562 (21)	0.597 (19)	0.655 (14)	0.567 (19)	0.589 (17)
Mod-Gtest	0.517 (21)	0.550 (22)	0.500 (21)	0.500 (18)	0.517 (19)	0.531 (22)	0.491 (22)	0.527 (22)	0.545 (20)	0.558 (19)
φPr-Gtest	0.450 (16)	0.417 (15)	0.417 (13)	0.383 (11)	0.300 (2)	0.648 (16)	0.675 (14)	0.665 (12)	0.701 (11)	0.768 (6)
Gtest	0.500 (19)	0.500 (21)	0.467 (17)	0.433 (14)	0.550 (21)	0.583 (19)	0.580 (21)	0.612 (19)	0.656 (14)	0.547 (21)
Exp-Conf	0.367 (7)	0.333 (3)	0.300 (1)*	0.283 (1)*	0.300 (2)	0.744 (6)	0.762 (3)	0.780 (2)	0.795 (1)*	0.780 (2)
Med-Conf	0.333 (4)	0.350 (5)	0.350 (8)	0.350 (8)	0.317 (7)	0.760 (4)	0.747 (5)	0.752 (7)	0.740 (8)	0.770 (5)
Mod-Conf	0.417 (12)	0.383 (10)	0.350 (8)	0.317 (4)	0.333 (9)	0.690 (11)	0.728 (8)	0.742 (8)	0.759 (5)	0.750 (9)
φPr-Conf	0.400 (9)	0.417 (15)	0.467 (17)	0.467 (16)	0.433 (13)	0.685 (12)	0.648 (17)	0.619 (18)	0.592 (17)	0.632 (13)
Conf	0.400 (9)	0.400 (13)	0.417 (13)	0.450 (15)	0.467 (15)	0.655 (15)	0.667 (15)	0.645 (15)	0.618 (15)	0.610 (16)
Exp-Freq	0.383 (8)	0.383 (10)	0.400 (11)	0.467 (16)	0.433 (13)	0.705 (7)	0.685 (13)	0.675 (11)	0.607 (16)	0.632 (13)
Freq	0.350 (5)	0.400 (13)	0.483 (20)	0.550 (21)	0.550 (21)	0.747 (5)	0.692 (12)	0.627 (16)	0.539 (21)	0.547 (21)

- DUG *with Confidence based discrimination scores*: the 5th group of methods are based upon G-test criterion, *i.e.*, "Exp-Conf", "Med-Conf", "Mod-Conf" and "φ Pr-Conf".
- *Simple Thresholding*: Another group methods we have compared are the feature selection methods for certain graphs. In order to get the certain graphs from the uncertain graphs in the dataset, we perform simple tresholding over the weights of the links to get the binary links. These baseline methods include: "Freq", "HISC", "Ratio", "Gtest" and "Conf", which correspond to the discrimination scores used in previous 5 groups separately.

LibSVM (60) with the linear kernel is used as the base classifier for all compared methods. The *min_sup* in the gSpan for ADHD, ADNI and HIV datasets are 20%, 40% and 40% respectively. Since the range of different discrimination functions can be extremely different. We set

TABLE XII. Results on the ADHD (Attention Deficit Hyperactivity Disorder) dataset with different number of features ($t = 100, \cdots, 500$). The results are reported as "average performance + (rank)".

	Error Rate ↓					F1 ↑				
Methods	$t = 100$	$t = 200$	$t = 300$	$t = 400$	$t = 500$	$t = 100$	$t = 200$	$t = 300$	$t = 400$	$t = 500$
Exp-HSIC	0.423 (10)	0.438 (13)	0.455 (14)	0.455 (11)	0.448(12)	0.593 (10)	0.564 (13)	0.543 (14)	0.547 (11)	0.549 (12)
Med-HSIC	0.420 (9)	0.405 (8)	0.413 (8)	0.448 (10)	0.433 (6)	0.569 (13)	0.597 (7)	0.593 (5)	0.549 (10)	0.562 (7)
Mod-HSIC	0.390 (4)	0.405 (8)	0.403 (4)	0.393 (1)*	0.410 (2)	0.614 (3)	0.599 (6)	0.596 (4)	0.594 (1)*	0.584 (2)
φPr-HSIC	0.432 (12)	0.470 (17)	0.475 (16)	0.513 (22)	0.503 (21)	0.597 (7)	0.563 (14)	0.554 (13)	0.508 (17)	0.525 (18)
*H*SIC	0.529 (22)	0.510 (20)	0.488 (17)	0.455 (11)	0.485 (17)	0.505 (22)	0.494(18)	0.498 (18)	0.538 (13)	0.526 (17)
Exp-Ratio	0.388 (3)	0.400 (5)	0.415 (10)	0.440 (8)	0.420 (4)	0.613 (4)	0.604 (5)	0.587 (9)	0.556 (8)	0.576 (4)
Med-Ratio	0.450 (16)	0.418 (11)	0.388 (1)*	0.428 (6)	0.410 (2)	0.554 (15)	0.586 (12)	0.619 (1)*	0.571 (5)	0.579 (3)
Mod-Ratio	0.400 (7)	0.370 (1)*	0.408 (5)	0.435 (7)	0.428 (5)	0.595 (8)	0.634 (1)*	0.591 (7)	0.558 (7)	0.560 (9)
φPr-Ratio	0.372 (1)*	0.430 (12)	0.410 (7)	0.415 (2)	0.408 (1)*	0.630 (1)*	0.589 (9)	0.590 (8)	0.591 (2)	0.589 (1)*
*R*atio	0.515 (20)	0.520 (21)	0.490 (18)	0.475 (17)	0.498 (19)	0.550 (16)	0.461 (22)	0.461 (21)	0.503 (19)	0.517 (20)
Exp-Gtest	0.393 (6)	0.403 (7)	0.413 (8)	0.420 (3)	0.435 (9)	0.610 (5)	0.588 (10)	0.582 (10)	0.586 (3)	0.563 (5)
Med-Gtest	0.437 (13)	0.400 (5)	0.408 (5)	0.420 (3)	0.453 (15)	0.559 (14)	0.590 (8)	0.600 (2)	0.580 (4)	0.551 (10)
Mod-Gtest	0.448 (15)	0.383 (4)	0.398 (2)	0.428 (5)	0.433 (6)	0.571 (12)	0.622 (4)	0.593 (5)	0.565 (6)	0.551 (10)
φPr-Gtest	0.450 (16)	0.445 (14)	0.443(12)	0.455 (11)	0.433 (6)	0.544 (19)	0.555 (16)	0.552 (12)	0.538 (13)	0.562 (7)
*G*test	0.440 (14)	0.505 (19)	0.501 (21)	0.486 (19)	0.471 (16)	0.542 (20)	0.492 (19)	0.490 (20)	0.499 (21)	0.534 (16)
Exp-Conf	0.405 (8)	0.415 (10)	0.453 (13)	0.455 (11)	0.448 (12)	0.595 (8)	0.587 (11)	0.539 (15)	0.543 (12)	0.535 (15)
Med-Conf	0.378 (2)	0.373 (2)	0.438 (11)	0.463 (15)	0.435 (9)	0.629 (2)	0.632 (2)	0.555 (11)	0.536 (15)	0.545 (13)
Mod-Conf	0.392 (5)	0.373 (2)	0.400 (3)	0.440 (8)	0.435 (9)	0.606 (6)	0.627 (3)	0.600 (2)	0.556 (8)	0.563 (5)
φPr-Conf	0.468 (19)	0.460 (15)	0.495 (20)	0.505 (21)	0.485 (17)	0.547 (18)	0.556 (15)	0.519 (16)	0.507 (18)	0.540 (14)
*C*onf	0.455 (18)	0.500 (18)	0.460 (15)	0.464 (16)	0.450 (14)	0.514 (21)	0.479 (20)	0.510 (17)	0.498 (22)	0.519 (19)
Exp-Freq	0.423 (10)	0.465 (16)	0.508 (22)	0.498 (20)	0.505 (22)	0.579 (11)	0.549 (17)	0.496 (19)	0.513 (16)	0.498 (22)
*F*req	0.515 (20)	0.520 (21)	0.490 (18)	0.475 (17)	0.498 (19)	0.550 (16)	0.461 (21)	0.461 (21)	0.503 (19)	0.517 (20)

the default φ for HSIC criterion, G-test score, frequency ratio and confidence as 0.03, 200, 1 and 0.5, respectively.

5.5.3 Performance on Uncertain Graph Classification

In our experiments, we first randomly sample 80% of the uncertain graphs as the training set, and the remaining graphs as the test set. This random sampling experiment was repeated 20 times. The average performances with the rank of each method are reported. The reason for using classification performances to evaluate the quality of subgraph features is that classification methods can usually achieve higher accuracy with features of better discriminative powers. We measure the classification performance by error rate and F1 score.

TABLE XIII. Results on the HIV (Human Immunodeficiency Virus) dataset with different number of features ($t = 100, \cdots, 500$). The results are reported as "average performance + (rank)".

	Error Rate ↓					F1 ↑				
Methods	$t = 100$	$t = 200$	$t = 300$	$t = 400$	$t = 500$	$t = 100$	$t = 200$	$t = 300$	$t = 400$	$t = 500$
Exp-HSIC	0.480 (15)	0.470 (10)	0.489 (12)	0.505 (16)	0.498 (13)	0.526 (13)	0.531 (8)	0.517 (11)	0.491 (14)	0.492 (13)
Med-HSIC	0.498 (17)	0.500 (18)	0.470 (7)	0.484 (11)	0.507 (16)	0.501 (18)	0.493 (18)	0.526 (8)	0.510 (10)	0.474 (16)
Mod-HSIC	0.502 (18)	0.489 (15)	0.482 (11)	0.498 (14)	0.500 (14)	0.501 (18)	0.501 (16)	0.495 (14)	0.481 (17)	0.467 (19)
φPr-HSIC	0.523 (19)	0.511 (19)	0.516 (18)	0.525 (19)	0.523 (20)	0.484 (20)	0.492 (19)	0.481 (16)	0.474 (19)	0.482 (14)
*H*SIC	0.464 (6)	0.495 (17)	0.566 (21)	0.500 (15)	0.505 (15)	0.526 (13)	0.460 (20)	0.405 (21)	0.489 (15)	0.471 (18)
Exp-Ratio	0.475 (13)	0.477 (11)	0.491 (13)	0.516 (18)	0.484 (8)	0.541 (8)	0.533 (7)	0.509 (13)	0.477 (18)	0.519 (8)
Med-Ratio	0.466 (8)	0.464 (8)	0.470 (7)	0.457 (5)	0.473 (6)	0.541 (8)	0.528 (9)	0.524 (9)	0.534 (6)	0.521 (6)
Mod-Ratio	0.450 (3)	0.452 (5)	0.466 (4)	0.480 (9)	0.484 (8)	0.558 (5)	0.547 (5)	0.528 (6)	0.509 (11)	0.500 (12)
φPr-Ratio	0.473 (11)	0.480 (12)	0.466 (4)	0.470 (8)	0.468 (5)	0.544 (7)	0.519 (13)	0.538 (5)	0.531 (7)	0.538 (5)
*R*atio	0.530 (21)	0.486 (13)	0.589 (22)	0.411 (1)*	0.520 (19)	0.456 (21)	0.495 (17)	0.376 (22)	0.562 (4)	0.443 (20)
Exp-Gtest	0.468 (9)	0.466 (9)	0.468 (6)	0.466 (7)	0.482 (7)	0.562 (4)	0.565 (4)	0.548 (4)	0.537 (5)	0.520 (7)
Med-Gtest	0.464 (6)	0.461 (7)	0.507 (17)	0.507 (17)	0.511 (17)	0.534 (11)	0.520 (11)	0.480 (17)	0.483 (16)	0.474 (16)
Mod-Gtest	0.477 (14)	0.486 (13)	0.475 (10)	0.491 (13)	0.489 (11)	0.529 (12)	0.507 (14)	0.523 (10)	0.497 (13)	0.501 (11)
φPr-Gtest	0.430 (1)*	0.420 (2)	0.425 (1)*	0.418 (2)	0.425 (2)	0.617 (1)*	0.633 (1)*	0.630 (1)*	0.637 (1)*	0.633 (1)*
*G*test	0.473 (11)	0.550 (21)	0.493 (14)	0.534 (20)	0.493 (12)	0.514 (16)	0.426 (22)	0.491 (15)	0.509 (11)	0.477 (15)
Exp-Conf	0.457 (4)	0.430 (4)	0.441 (2)	0.443 (4)	0.441 (3)	0.576 (3)	0.590 (2)	0.572 (2)	0.570 (3)	0.573 (4)
Med-Conf	0.445 (2)	0.427 (3)	0.441 (2)	0.441 (3)	0.443 (4)	0.579 (2)	0.588 (3)	0.572 (2)	0.579 (2)	0.574 (3)
Mod-Conf	0.457 (4)	0.455 (6)	0.473 (9)	0.482 (10)	0.484 (8)	0.556 (6)	0.545 (6)	0.527 (7)	0.518 (9)	0.508 (9)
φPr-Conf	0.534 (22)	0.552 (22)	0.545 (19)	0.548 (21)	0.541 (22)	0.454 (22)	0.443 (21)	0.444 (20)	0.443 (22)	0.438 (21)
*C*onf	0.468 (9)	0.416 (1)*	0.502 (15)	0.489 (12)	0.339 (1)*	0.515 (15)	0.528 (9)	0.468 (19)	0.462 (20)	0.621 (2)
Exp-Freq	0.525 (20)	0.520 (20)	0.548 (20)	0.550 (22)	0.527 (21)	0.503 (17)	0.520 (11)	0.473 (18)	0.457 (21)	0.423 (22)
*F*req	0.489 (16)	0.489 (15)	0.502 (15)	0.461 (6)	0.514 (18)	0.535 (10)	0.505 (15)	0.517 (11)	0.520 (8)	0.502 (10)

Table XII and Table XI show the evaluation results in terms of classification error rates and F1 scores with different number of selected subgraph features ($t = 100, \cdots, 500$). The results of each method are shown with average performance values and their ranks among all the other methods. Values with $*$ stand for the best performance for the corresponding evaluation criterion. It is worth noting that the neuroimaging tasks are generally very hard to predict very accurately. According to a global competition on ADHD dataset[1], the average performance of all winning teams is about 8% over the prediction accuracy of chance (i.e., randomly assigning

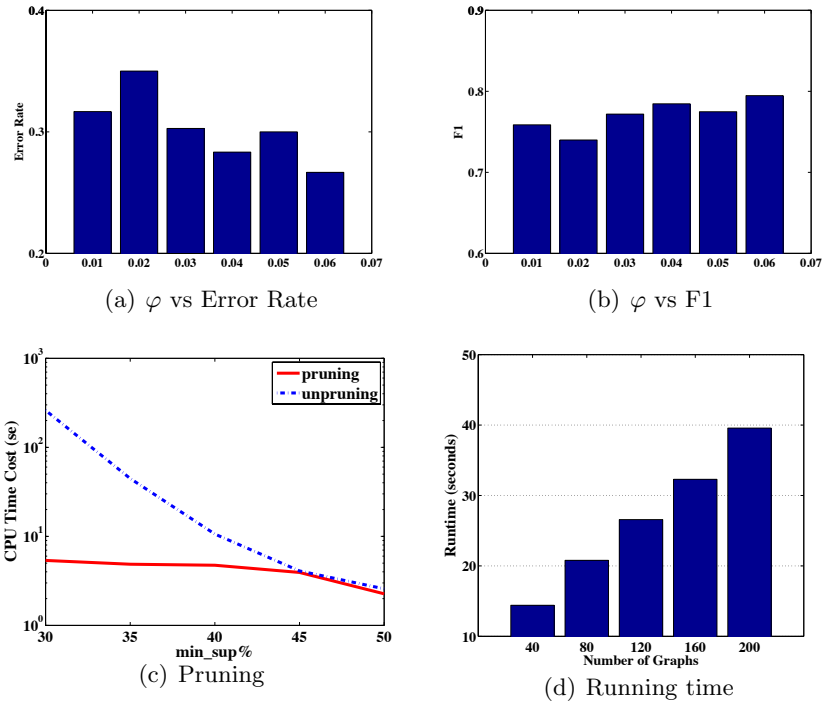
[1]http://www.childmind.org/en/posts/press-releases/2011-10-12-johns-hopkins-team-wins-adhd-200-competition

diagnoses). Thus in neuroimaging tasks, it is very hard for classification algorithms to achieve even moderate error rates. And in ADHD dataset, the best performance that DUG can achieve is with error rate 37%, which is 13% improvement over the prediction error rate of chance.

We find that our discriminative subgraph mining method with different settings outperforms the baseline method (Exp-Freq) for frequent subgraph mining, which selects subgraph features based upon expected frequencies in the uncertain graph dataset. This is because that frequent subgraph features in uncertain graph dataset may not be relevant to the classification task.

Moreover, we can see that almost all the DUG methods outperform the simple thresholding methods which directly convert the uncertain graphs into certain graphs and then use different discimination functions to select subgraph features. This is because that simply converting uncertain graphs into certain graph can loss the uncertainty information about the linkage structures of the graphs, thus the classification performances on certain graphs are not as good as the performance of uncertain graphs.

A third observation is that the performance of each method on different dataset can be quite different. However, the best methods that consistently outperforms other methods in all datasets are Med-Conf and φ-Pr-Ratio. They both have their advantages in different perspectives. Med-Conf method has one less parameter than that of φ-Pr-Ratio. φ-Pr-Ratio method has an additional subgraph pruning bound compared to Med-Conf method, which can be important for datasets with even larger graphs.



(a) φ vs Error Rate (b) φ vs F1

(c) Pruning (d) Running time

Figure 34. Parameter Studies (ADNI dataset).

5.5.4 Influence of Parameter

In the φ-Pr based methods, there is an additional threshold parameter than the other methods. In 34(a) and 34(b), we tested the φ-Pr-HSIC with φ values among $\{0.01, 0.02, \cdots 0.06\}$ separately. We can see that the method is not sensitive to the parameter φ. Generally, the performance of φ-Pr-HSIC with default setting ($\varphi = 0.03$) is pretty good. If we try to optimize the selection of φ value, the accuracy can be even better.

We also compare DUG models with and without pruning in the subgraph search space as summarized in 34(c). The CPU time with different *min_sup* for Exp-HSIC in ADNI dataset

is reported. DUG can improve the efficiencies by pruning the subgraph search space. In other datasets DUG shows similar trends. 34(d) shows the running time for mod-HISC with different number of graphs in the dataset. In addition to the dynamic programming method we used in DUG, we also find that the brute-force searching method that enumerates all possible worlds of the uncertain graph dataset cannot work on small datasets with even 40 graphs. The running time of DUG scales almost linearly with the number of graphs in the dataset. Althought the dynamic programming process of DUG is $O(n^2)$, which is quadratic to the number of graphs in the dataset. However, in the ADHD dataset, the main computational cost of DUG algorithm is for the subgraph enumeration step, which is linear to the number of the graphs in the dataset. In cases of even larger datasets, the dynamic programming process could eventually dominant the computational cost. In these cases, the divide-and-conquer method in (72) could be used to further optimize the computational cost.

5.6 Conclusion

In this chapter, we studied the problem of discriminative subgraph feature selection for uncertain graph classification. We proposed a general framework, called DUG, for finding discriminative subgraph feature in uncertain graphs based upon various statistical measures. The probability distributions of the scoring function are efficiently computed based on dynamic programming.

CHAPTER 6

CONCLUSION

In this thesis, we have studied four different settings of graph classification: multi-label setting, semi-supervised setting, active learning setting, and uncertain graph setting.

In the multi-label setting, we studied the problem of multi-label feature selection for graph classification and propose a novel solution, called gMLC, to efficiently search for optimal subgraph features for graph objects with multiple labels. Different from existing feature selection methods in vector spaces which assume the feature set is given, we perform multi-label feature selection for graph data in a progressive way together with the subgraph feature mining process. We derived an evaluation criterion to estimate the dependence between subgraph features and multiple labels of graphs. Then a branch-and-bound algorithm was proposed to efficiently search for optimal subgraph features by judiciously pruning the subgraph search space using multiple labels. Empirical studies demonstrated that our feature selection approach can effectively boost multi-label graph classification performances and is more efficient by pruning the subgraph search space using multiple labels.

In the semi-supervised setting, we studied the problem of semi-supervised feature selection for graph classification and propose a novel solution, called gSSC, to efficiently search for optimal subgraph features with labeled and unlabeled graphs. Different from existing feature selection methods in vector spaces which assume the feature set is given, we perform semi-supervised feature selection for graph data in a progressive way together with the subgraph

feature mining process. We derived a feature evaluation criterion, named gSemi, to estimate the usefulness of subgraph features based upon both labeled and unlabeled graphs. Then we proposed a branch-and-bound algorithm to efficiently search for optimal subgraph features by judiciously pruning the subgraph search space. Empirical studies on several real-world tasks demonstrated that our semi-supervised feature selection approach can effectively boost graph classification performances with semi-supervised feature selection and is very efficient by pruning the subgraph search space using both labeled and unlabeled graphs.

In the active learning setting, we addressed the problem of how to select the most important graph to query for the label. This problem is challenging and different from conventional active learning problems because there is no predefined feature vector. The active sample selection problem and the feature selection problem are *correlated* for graph data. Before we can solve the active sample selection problem, we need to find a set of optimal subgraph features. We demonstrated how one can simultaneously estimate the usefulness of a query graph and a set of subgraph features. The idea is to maximize the dependency between subgraph features and graph labels using an active learning framework. We proposed a branch-and-bound algorithm to search for the optimal query graph and optimal features simultaneously. Empirical studies on nine real-world tasks demonstrated that the proposed method can obtain better accuracy on graph data than alternative approaches.

In the uncertain graph setting, we studied the problem of discriminative subgraph feature selection from uncertain graphs. This problem is challenging and different from conventional subgraph mining problems because both the structure of the graph objects and the discrimi-

nation score of each subgraph feature are uncertain. To address these challenges, we proposed a novel discriminative subgraph feature selection method, DUG, which can find discriminative subgraph features in uncertain graphs based upon different statistical measures including expectation, median, mode and φ-probability. We first computed the probability distribution of the discrimination scores for each subgraph feature based on dynamic programming. Then a branch-and-bound algorithm was proposed to search for discriminative subgraphs efficiently. Extensive experiments on various neuroimaging applications (*i.e.*, Alzheimer's Disease, ADHD and HIV) have been performed to analyze the gain in performance by taking into account structural uncertainties in identifying discriminative subgraph features for graph classification.

APPENDICES

.1 ACM Copyright Letter

"Authors can reuse any portion of their own work in a new work of their own (and no fee is expected) as long as a citation and DOI pointer to the Version of Record in the ACM Digital Library are included.

Contributing complete papers to any edited collection of reprints for which the author is not the editor, requires permission and usually a republication fee. Authors can include partial or complete papers of their own (and no fee is expected) in a dissertation as long as citations and DOI pointers to the Versions of Record in the ACM Digital Library are included. Authors can use any portion of their own work in presentations and in the classroom (and no fee is expected)." [1]

.2 Springer Copyright Letter

"The author retains the right to use his/her article for his/her further scientific career by including the final published journal article in other publications such as dissertations and postdoctoral qualifications provided acknowledgement is given to the original source of publication."[2]

---

[1] http://authors.acm.org/main.html

[2] http://www.springer.com/?SGWID=4-102-45-69724-0

CITED LITERATURE

1. Kong, X. and Yu, P. S.: gmlc: a multi-label feature selection framework for graph classification. Knowledge and Information Systems, 2011. doi:10.1007/s10115-011-0407-3.

2. Kong, X. and Yu, P. S.: Semi-supervised feature selection for graph classification. In KDD, 2010. doi:10.1145/1835804.1835905.

3. Kong, X. and Yu, P. S.: Dual active feature and sample selection for graph classification. In KDD, 2011. doi:10.1145/2020408.2020511.

4. Kong, X., Yu, P. S., Wang, X., and Ragin, A. B.: Discriminative feature selection for uncertain graph classification. In SDM, 2013.

5. Huang, S., Li, J., Sun, L., Ye, J., Chen, K., and Wu, T.: Learning brain connectivity of azheimer's disease from neuroimaging data. In NIPS, pages 808–816, 2009.

6. Huang, S., Li, J., Ye, J., Fleisher, A., Wu, T., Chen, K., and Reiman, E.: Learning brain connectivity of alzheimer's disease by exploratory graphical models. NeuroImage, 50:935–949, 2010.

7. Huang, S., Li, J., Ye, J., Fleisher, A., Chen, K., Wu, T., and Reiman, E.: Brain effective connectivity modeling for alzheimer's disease by sparse gaussian bayesian network. In KDD, pages 931–939, 2011.

8. Zhou, J., Yuan, L., Liu, J., and Ye, J.: A multi-task learning formulation for predicting disease progression. In KDD, pages 814–822, 2011.

9. Chen, C., Yan, X., Zhu, F., Han, J., and Yu, P.: Graph OLAP: a multi-dimensional framework for graph data analysis. Knowledge and Information Systems, 21(1):41–63, 2009.

10. Tasourakakis, U. K. C. and Faloutsos, C.: Pegasus: mining peta-scale graphs. Knowledge and Information Systems, pages 1–23, 2010.

11. Jia, Y., Tao, J., and Huan, J.: An efficient graph-mining method for complicated and noisy data with real-world applications. Knowledge and Information Systems, pages 1–25, 2011.

12. Ying, X. and Wu, X.: On link privacy in randomizing social networks. Knowledge and Information Systems, pages 1–19, 2010.

13. Yan, X., Cheng, H., Han, J., and Yu, P.: Mining significant graph patterns by leap search. In SIGMOD, pages 433–444, 2008.

14. Thoma, M., Cheng, H., Gretton, A., Han, J., Kriegel, H., Smola, A., Song, L., Yu, P., Yan, X., and Borgwardt, K.: Near-optimal supervised feature selection among frequent subgraphs. In SDM, pages 1075–1086, 2009.

15. Fei, H. and Huan, J.: Boosting with structure information in the functional space: an application to graph classification. In Proceedings of the 16th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 643–652, Washington, DC, 2010.

16. Zou, Z., Gao, H., and Li, J.: Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. In KDD, pages 633–642, 2010.

17. McCallum, A.: Multi-label text classification with a mixture model trained by EM. In Working Notes of the AAAI'99 Workshop on Text Learning, Orlando, FL, 1999.

18. Schapire, R. E. and Singer, Y.: Boostexter: a boosting-based system for text categorization. Machine Learning, 39(2-3):135–168, 2000.

19. Elisseeff, A. and Weston, J.: A kernel method for multi-labelled classification. In Advances in Neural Information Processing Systems 14, pages 681–687. 2002.

20. Boutell, M. R., J. Luo, Shen, X., and Brown, C. M.: Learning multi-label scene classification. Pattern Recognition, 37(9):1757–1771, 2004.

21. Kudo, T., Maeda, E., and Matsumoto, Y.: An application of boosting to graph classification. In NIPS, pages 729–736, 2005.

22. Zhang, M.-L. and Zhou, Z.-H.: Ml-knn: A lazy learning approach to multi-label learning. Pattern Recognition, 40(7):2038–2048, 2007.

23. Ueda, N. and Saito, K.: Parametric mixture models for multi-labeled text. In Advances in Neural Information Processing Systems 13, pages 721–728. 2003.

24. Comité, F. D., Gilleron, R., and Tommasi, M.: Learning multi-label altenating decision tree from texts and data. In Proceedings of the 3rd International Conference on Machine Learning and Data Mining in Pattern Recognition, pages 35–49, Leipzig, Germany, 2003.

25. Godbole, S. and Sarawagi, S.: Discriminative methods for multi-labeled classification. In Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 22–30, Sydney, Australia, 2004.

26. Kazawa, H., Izumitani, T., Taira, H., and Maeda, E.: Maximal margin labeling for multi-topic text categorization. In Advances in Neural Information Processing Systems 15, pages 649–656. 2005.

27. G. Tsoumakas, I. V.: Random k-labelsets: An ensemble method for multilabel classification. In Proceedings of the 18th European Conference on Machine Learning, pages 406–417, Warsaw, Poland, 2007.

28. Yan, X. and Han, J.: gSpan: Graph-based substructure pattern mining. In ICDM, pages 721–724, 2002.

29. Inokuchi, A., Washio, T., and Motoda, H.: An apriori-based algorithm for mining frequent substructures from graph data. In ECML/PKDD, pages 13–23, 2000.

30. Kuramochi, M. and Karypis, G.: Frequent subgraph discovery. In ICDM, pages 313–320, 2001.

31. Borgelt, C. and Berthold, M.: Mining molecular fragments: Finding relevant substructures of molecules. In ICDM, pages 211–218, 2002.

32. Huan, J., Wang, W., and Prins, J.: Efficient mining of frequent subgraph in the presence of isomorphism. In ICDM, pages 549–552, 2003.

33. Nijssen, S. and Kok, J.: A quickstart in frequent structure mining can make a difference. In KDD, pages 647–652, 2004.

34. Kong, X. and Yu, P. S.: Semi-supervised feature selection for graph classification. In KDD, pages 793–802, 2010.

35. Borgwardt, K. M.: Graph Kernels. Doctoral dissertation, Ludwig-Maximilians-University Munich, 2007.

36. Zhang, Y. and Zhou, Z.-H.: Multi-label dimensionality reduction via dependency maximization. In Proceedings of the 23rd AAAI Conference on Artificial Intelligence, pages 1053–1055, Chicago, IL, 2008.

37. Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B.: Measuring statistical dependence with Hilbert-Schmidt norms. In ALT, pages 63–77, Singapore, 2005.

38. Helma, C., King, R., Kramer, S., and Srinivasan, A.: The predictive toxicology challenge 2000-2001. Bioinformatics, 17(1):107–108, 2001.

39. Kashima, H., Tsuda, K., and Inokuchi, A.: Marginalized kernels between labeled graphs. In ICML, pages 321–328, 2003.

40. Bar-Hillel, A., Hertz, T., Shental, N., and Weinshall, D.: Learning a mahalanobis metric from equivalence constraints. Journal of Machine Learning Research, 6:937–965, 2005.

41. Tang, W. and Zhong, S.: Pairwise constraints-guided dimensionality reduction. In SIAM International Conference on Data Mining Workshop on Feature Selection for Data Mining, Bethesda, MD, 2006.

42. Zhao, Z. and Liu, H.: Semi-supervised feature selection via spectral analysis. In Proceedings of the SIAM International Conference on Data Mining, pages 641–646, Minneapolis, MN, 2007.

43. Ren, J., Qiu, Z., Fan, W., Cheng, H., and Yu, P. S.: Forword semi-supervised feature selection. In Proceedings of the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 970–976, Osaka, Japan, 2008.

44. Mardia, K. V., Kent, J. T., and Bibby, J. M.: Multivariate Analysis. San Diego, CA, Academic Press, 1980.

45. Yan, X., Yu, P. S., and Han, J.: Graph indexing based on discriminative frequent struture analysis. ACM Transactions on Database Systems, 30(4):960–993, 2005.

46. Tong, S. and Koller, D.: Support vector machine active learning with applications to text classification. In ICML, pages 999–1006, 2000.

47. Yang, B., Sun, J., Wang, T., and Chen, Z.: Effective multi-label active learning for text classification. In KDD, pages 917–926, 2009.

48. Jin, N., Young, C., and Wang, W.: GAIA: graph classification using evolutionary computation. In SIGMOD, pages 879–890, 2010.

49. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

50. Balcan, M. F., Broder, A. Z., and Zhang, T.: Margin based active learning. In COLT, pages 35–50, 2007.

51. Dagan, I. and Engenlson, S. P.: Committee-based sampling for training probabilistic classifiers. In ICML, pages 150–157, 1995.

52. Freund, Y., H. S. Seung, E. S., and Tishby, N.: Selective sampling using the query by committee algorithm. Machine Learning, 28(2-3):133–168, 1997.

53. Opper, M., Seung, H. S., and Sompolinsky, H.: Query by committee. In COLT, pages 287–294, 1992.

54. Nguyen, H. T. and Smeulders, A. W. M. .: Active learning using pre-cluster. In ICML, pages 623–630, 2004.

55. Dasgupta, S. and Hsu, D.: Hierarchical sampling for active learning. In ICML, pages 208–215, 2008.

56. Yu, K., Bi, J., and Tresp, V.: Active learning via transductive experimental design. In ICML, pages 1081–1088, 2006.

57. Donmez, P., Carbonell, J. G., and Bennett, P. N.: Dual strategy active learning. In ECML, pages 116–127, 2007.

58. Huang, S.-J., Jin, R., and Zhou, Z.-H.: Active learning by querying informative and representative examples. In NIPS. 2011.

59. Fan, W., Zhang, K., Cheng, H., Gao, J., Yan, X., Han, J., Yu, P. S., and Verscheure, O.: Direct mining of discriminative and essential frequent patterns via model-based search tree. In KDD, pages 230–238, 2008.

60. Chang, C.-C. and Lin, C.-J.: LIBSVM: a library for support vector machines, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

61. Bullmore, E. and Sporns, O.: Complex brain networks: graph theoretical analysis of structural and functional systems. Nature Reviews Neuroscience, 10(3):186–198, 2009.

62. Wang, X., Foryt, P., Ochs, R., Chung, J., Wu, Y., Parrish, T., and Ragin, A.: Abnormalities in resting-state functional connectivity in early human immunodeficiency virus infection. Brain Connectivity, 1(3):207, 2011.

63. Zou, Z., Li, J., Gao, H., and Zhang, S.: Frequent subgraph pattern mining on uncertain graph data. In CIKM, pages 583–592, 2009.

64. Zou, Z., Li, J., Gao, H., and Zhang, S.: Mining frequent subgraph patterns from uncertain graph data. IEEE Transactions on Knowledge and Data Engineering, 22(9):1203–1218, 2010.

65. Papapetrou, O., Ioannou, E., and Skoutas, D.: Efficient discovery of frequent subgraph patterns in uncertain graph databases. In EDBT, pages 355–366, 2011.

66. Jin, R., Liu, L., and Aggarwal, C.: Discovering highly reliable subgraphs in uncertain graphs. In KDD, pages 992–1000, 2011.

67. Potamias, M., Bonchi, F., Gionis, A., and Kollios, G.: k-nearest neighbors in uncertain graphs. In VLDB, pages 997–1008, 2010.

68. Yuan, Y., Wang, G., Wang, H., and Chen, L.: Efficient subgraph search over large uncertain graphs. In VLDB, pages 876–886, 2011.

69. Jin, N. and Wang, W.: LTS: Discriminative subgraph mining by learning from search history. In ICDE, pages 207–218, 2011.

70. Gao, C. and Wang, J.: Direct mining of discriminative patterns for classifying uncertain data. In KDD, pages 861–870, 2010.

71. Kong, X., Fan, W., and Yu, P. S.: Dual active feature and sample selection for graph classification. In KDD, pages 654–662, 2011.

72. Sun, L., Cheng, R., Cheung, D., and Cheng, J.: Mining uncertain data with probabilistic guarantees. In KDD, pages 273–282, 2010.

VITA

EDUCATION

Nanjing University, M.S. in Computer Science 2009

Nanjing University, B.Sc. in Computer Science 2006

PUBLICATIONS

Conference Papers

[C26] **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." *S*IAM International Conference on Data Mining (**SDM**), 2014

[C25] Ning Yang, **Xiangnan Kong**, Fengjiao Wang, and Philip S. Yu. "When and Where: Predicting Human Movements Based on Social Spatial-Temporal Events." *S*IAM International Conference on Data Mining (**SDM**), 2014

[C24] Lifang He, **Xiangnan Kong**, Philip S. Yu, Ann B. Ragin, Zhifeng Hao, and Xiaowei Yang. "DuSK: A Dual Structure-preserving Kernel for Supervised Tensor Learning with Applications to Neuroimages." *S*IAM International Conference on Data Mining (**SDM**), 2014

[C23] Jiawei Zhang, **Xiangnan Kong**, and Philip S. Yu. "Transferring Heterogeneous Links across Location-Based Social Networks." *A*CM International Conference on Web Search and Data Mining (**WSDM**), 2014

[C22] Chun-Ta Lu, Sihong Xie, **Xiangnan Kong**, and Philip S. Yu. "Inferring the Impacts of Social Media on Crowdfunding." *A*CM International Conference on Web Search and Data Mining (**WSDM**), 2014

[C21] **X**iangnan Kong, Bokai Cao and Philip S. Yu. "Multi-label Classification by Mining Instance and Label Correlation from Heterogeneous Information Network." In Proc. *A*CM SIGKDD Conference on Knowledge Discovery and Data Mining (**KDD**), 2013

[C20] Sihong Xie, **Xiangnan Kong**, Jing Gao, Wei Fan, and Philip S. Yu. "Multilabel Consensus Classification." In Proc. *I*EEE International Conference on Data Mining (**ICDM**), 2013.

[C19] Jiawei Zhang, **Xiangnan Kong**, and Philip S. Yu. "Predicting Social Links for New Users across Aligned Heterogeneous Social Networks." In Proc. *I*EEE International Conference on Data Mining (**ICDM**), 2013.

[C18] **Xiangnan Kong**, Jiawei Zhang and Philip S. Yu. "Inferring Anchor Links across Heterogeneous Social Networks." In Proc. *A*CM International Conference on Information and Knowledge Management (**CIKM**), 2013.

[C17] Shuyang Lin, **Xiangnan Kong** and Philip S. Yu. "Predicting Trends in Social Networks via Dynamic Activeness Model." In Proc. *A*CM International Conference on Information and Knowledge Management (**CIKM**), 2013.

[C16] **Xiangnan Kong**, Philip S. Yu, Xue Wang and Ann B. Ragin. "Discriminative Feature Selection for Uncertain Graph Classification." In Proc. *S*IAM International Conference on Data Mining (**SDM**), 2013

[C15] Li-Jia Li, **Xiangnan Kong**, and Philip S. Yu. "Visual Recognition by Exploiting Latent Social Links in Image Collections." In Proc. *I*nternational Conference on MultiMedia Modeling (**MMM**), 2013.

[C14] **Xiangnan Kong**, Philip S. Yu, Ying Ding, and David Wild. "Meta Path-Based Collective Classification in Heterogeneous Information Networks." In Proc. *A*CM International Conference on Information and Knowledge Management (**CIKM**), 2012.

[C13] Guanqun Wang, **Xiangnan Kong**, Philip S. Yu, Quanyuan Wu, Yan Jia and Chuan Li. "Community Dection in Incomplete Information Networks." In Proc. *I*nternational World Wide Web Conferences (**WWW**), 2012. (12% acceptance)

[C12] Chuan Shi, **Xiangnan Kong**, Philip S. Yu, and Sihong Xie. "Relevance Search in Heterogeneous Networks." In Proc. *I*nternational Conference on Extending Database Technology (**EDBT**), 2012. (22.5% acceptance)

[C11] Chuan Shi, Chong Zhou, **Xiangnan Kong**, Philip S. Yu, and Gang Liu. "HeteRecom: A Semantic Recommendation System in Heterogeneous Networks." In Proc. *A*CM SIGKDD Conference on Knowledge Discovery and Data Mining (**KDD**), 2012 (System Demo)

[C10] Chuan Shi, **Xiangnan Kong** and Philip S. Yu. "Multi-Objective Multi-Label Classification." In Proc. *S*IAM International Conference on Data Mining (**SDM**), 2012 (Oral Presentation, 14.6% acceptance)

[C9] Xiaoxiao Shi, **Xiangnan Kong** and Philip S. Yu. "Transfer Significant Subgraphs across Graph Databases." In Proc. *S*IAM International Conference on Data Mining (**SDM**), 2012 (Oral Presentation, 14.6% acceptance)

[C8] Erjia Yan, Ying Ding, and **Xiangnan Kong**. "Monitoring knowledge flow through scholarly networks." In Proc. *A*merican Society for Information Science and Technology (ASIS&T), 2012

[C7] **Xiangnan Kong** and Philip S. Yu. "An Ensemble-based Approach to Fast Classification of Multi-label Data Streams." In Proc. *I*EEE International Conference on Collaborative Computing: Networking, Application and Worksharing (CoCom), 2011 (Invited Paper)

[C6] Yuchen Zhao, **Xiangnan Kong** and Philip S. Yu. "Positive and Unlabeled Learning for Graph Classification." In Proc. *I*EEE International Conference on Data Mining (**ICDM**), 2011 (Full Paper, 12.3% acceptance)

[C5] Chuan Shi, **Xiangnan Kong** and Philip S. Yu. "Multi-label Ensemble Learning." In Proc. *t*he European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (**ECMLPKDD**), 2011 (Oral Presentation, 20% acceptance)

[C4] **Xiangnan Kong** and Philip S. Yu. "Dual Active Feature and Sample Selection for Graph Classification." In Proc. *A*CM SIGKDD Conference on Knowledge Discovery and Data Mining (**KDD**), 2011 (Oral Presentation, 7.8% acceptance)

[C3] **Xiangnan Kong**, Xiaoxiao Shi and Philip S. Yu. "Multi-Label Collective Classification." In Proc. *S*IAM International Conference on Data Mining (**SDM**), 2011 (25.07% acceptance)

[C2] **Xiangnan Kong** and Philip S. Yu. "Multi-Label Feature Selection for Graph Classification." In Proc. *I*EEE International Conference on Data Mining (**ICDM**), 2010 (Full Paper, 9.03% acceptance)

[C1] **Xiangnan Kong** and Philip S. Yu. "Semi-Supervised Feature Selection for Graph Classification." In Proc. *A*CM SIGKDD Conference on Knowledge Discovery and Data Mining (**KDD**), 2010 (Full Paper, 13.3% acceptance)

Journal Papers

[J5] Chuan Shi, **Xiangnan Kong**, Yue Huang, Philip S. Yu, and Bin Wu. "HeteSim: A General Framework for Relevance Measure in Heterogeneous Networks" *I*EEE Trans. Knowledge and Data Engineering (**TKDE**), 2013

[J4] Chuan Shi, **Xiangnan Kong**, Di Fu, Philip S. Yu and Bai Wang. "Multi-label classification based on Multi-Objective Optimization." *A*CM Trans. Intelligent Systems and Technology (**TIST**), 2013

[J3] **Xiangnan Kong**, Michael Ng and Zhi-Hua Zhou. "Transductive Multi-Label Learning via Label Set Propagation." *I*EEE Trans. Knowledge and Data Engineering (**TKDE**), 2012

[J2] **Xiangnan Kong** and Philip S. Yu. "gMLC: a Multi-label Feature Selection Framework for Graph Classification." *K*nowledge and Information Systems (**KAIS**), 2011

[J1] **Xiangnan Kong**, Ming Li, Yuan Jiang and Zhi-Hua Zhou. "A Transductive Multi-Label Classification Method for Weak Labeling." *J*ournal of Computer Research and Development, 47(8):1392-1399, 2010

Book Chapter

[B2] **Xiangnan Kong**, and Philip Yu. "Graph Classification in Heterogeneous Networks." Book Chapter in *E*ncyclopedia of Social Networks and Mining (**ESNAM**).

[B1] Charu C. Aggarwal, **Xiangnan Kong**, Quanquan Gu, Jiawei Han, and Philip Yu. "Active Learning: A Survey." Book Chapter in *D*ata Classification: Algorithms and Applications.