EXPLOITING MULTIPLE DATA SOURCES FOR NETWORK MINING

By

Prakash Mandayam Comar

A DISSERTATION

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

Computer Science - Doctor of Philosophy

2013

ABSTRACT

EXPLOITING MULTIPLE DATA SOURCES FOR NETWORK MINING

By

Prakash Mandayam Comar

Network mining is an active research area with application to diverse fields including computer science, social science, and biological sciences. However, previous studies have focused mostly on developing algorithms for mining data from a single network. Such algorithms are susceptible to imperfections in the network data such as noisy links and node attribute values. The focus of this thesis is on exploiting multiple data sources to enhance the performance of network mining algorithms for community detection, node classification and link prediction tasks.

The first contribution of this thesis is the development of a joint matrix factorization framework for mining multiple networks. The framework offers a principled way to perform community detection simultaneously across multiple related networks. It is also highly flexible, allowing the link structure, node attributes, and any prior knowledge about the relationship between communities in different networks to be seamlessly integrated under a unified formulation. The framework is then extended to a multi-task learning setting where one could perform community detection on one network and node classification on the other.

Multi-task learning is natural for networks considering the intimate relation between the link structure and node attributes of the networks. However, designing a framework for multi-task network learning requires a joint objective function that can be used for various network mining tasks while accommodating some of the existing objective functions (such as the well-known modularity measure for community detection). As second contribution,

this thesis presents a novel cost-sensitive loss function that enables the joint learning for link prediction and community detection on one or more networks. The loss function addresses the class skewness and degree skewness problems inherent in most link prediction tasks. A formal proof is provided to show the equivalence between the proposed loss function and the modularity measure used in community detection. To enhance the scalability of the approach, a divide and conquer scheme was developed where the learning algorithm is applied to smaller partitions of a network and their results are systematically combined using the boosting framework.

Acquiring reliable labels is crucial for network learning tasks such as link prediction and node classification. While for the most part the labels can be gleaned from the network itself, they are often incomplete and noisy, thus requiring alternative mechanism to solicit more label information. This thesis explores the viability of using crowdsourcing technology as an external source for obtaining additional labeled data for network mining tasks. Adopting crowdsourcing for network data is non-trivial due to the difficulty in designing a human intelligence task (HIT) that can be easily handled by non-experts (i.e., the *crowd*). To overcome this problem, this thesis proposes an approach for transforming network data into a set of images that can be easily labeled by non-experts. The conditions under which the transformation preserves the original network data was also examined. To the best of our knowledge, this is the first study to examine the use of crowdsourcing for acquiring labels in network learning tasks.

This thesis is a step forward towards resolving some of the fundamental challenges in performing multi-source network mining. Though the methods described in this thesis were designed for network mining, some of them (e.g., methodology to transform network data into image data) are applicable to non-network learning problems.

ACKNOWLEDGMENTS

This PhD dissertation is an end product of a very long delightful journey during which I was helped by several wonderful personalities, many of whom have left a lasting impression in my mind. I would like to sincerely acknowledge each of their contribution in shaping this dissertation.

With at most sincerity and gratitude, I thank my advisor Dr. Pang-Ning Tan for all the technical guidance, support and encouragement he showered on me. He generously funded my PhD programme with assistantships and internship opportunities. He was always very approachable, kind and encouraged me to pursue problems of my liking and interest. He always stressed on the importance of mathematical representation and analysis of problems, instead of solving by heuristics and intuition. The data mining course offered by Dr. Tan, the biweekly meetings, seminars, paper reviews and generous funding for conference travel has kept me up to date with most recent technical advances in my field of research. In all he made my stay here in Michigan State University very comfortable, memorable and enlightening. I will certainly cherish my leanings from his association for long time to come.

Next, I would like to thank my PhD committee members, Prof. Anil K Jain, Dr. Rong Jin and Prof. Ramamoorthi for their support and guidance throughout my PhD programme. Dr. Jain took special interest in my work and advised me to pursue research, associating social networks with crowdsourcing. I was very delighted with every meeting and conversation I had with him and Chapter 6 of this dissertation is result of the impetus given by him. I would also like to specially thank Dr. Jin for shaping my thinking during the weekly seminar on convex optimization in fall 2010. His machine learning course and seminars had influential impact on my research work. Finally I thank Prof. Ramamoorthi for being instrumental

in me joining Michigan State University. It was great honor to have worked with each of my PhD committee member and I sincerely thank them for all the help and guidance they offered to me. I earnestly look forward to continue my research collaborations with each of my committee member in the future.

Further, I heartily thank my peer group here in LINKS lab Lei Liu, Zubin Abraham, Jianpeng, Shai and Xi, specially for their help in manual labeling of images which otherwise would have been a very costly and time consuming endeavor. I specially thank my colleague Meherdad Mahdavi for keeping my lab hours interesting and informative with his update on recent technical happenings in optimization field. My special thanks to our department secretaries Linda Moore and Norma Teague for helping with all the administrative and travel related work. My sincere thanks to all my friends and roommates for keeping my stay in East Lansing a happy and fun filled affair.

Finally I thank my parents for the unconditional love, support and encouragement they showered on me throughout my life. I dedicate this work to Sri Sathya Sai, who has been my inspiration and guide all though my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

Chapter 1

Introduction

The rapid growth of online social media services such as Facebook , Twitter and Wikipedia has triggered a wave of interest in applying data mining and machine learning techniques to the study of complex networks. The field of social network analysis (SNA) has focused on a variety of issues, from inferring the formation of links in a social network to understanding how social phenomena such as homophily, social influence, and communities emerge from the interactions between individuals in a network. Significant progress was achieved in the 1990s following the seminal works of Watts et al. [104] and Barabasi et al. [8], who showed that the structural properties of many social, biological, and physical networks can be characterized by the same mathematical principles. This has led to increasing research into the development and application of *network mining* techniques to other areas beyond social science, including systems biology (e.g., for modeling proteins and gene interaction networks) and geo-sciences (e.g., for modeling teleconnections among climate variables and ecological processes).

A great deal of research in network mining has focused on the development of algorithms for mining data from a single network. This includes algorithms for solving network link prediction, node classification, and community detection tasks. As more diverse sources of network data becomes available, the need for *multi-source network mining* has grown in recent years. However, developing a robust algorithm that can effectively combine information from other networks is a challenge. The algorithm needs to be applied not only to one

network, but also simultaneously to other related networks. This requires a flexible computational framework that can utilize potentially noisy information from other sources and produce consistent solutions across all networks. Furthermore, the learning tasks applied to the multiple networks may not be the same. For example, one might be interested in predicting the formation of links in a network based on the community structure found in another network. Previous works, which are mostly limited to performing the same task on one or more networks, must therefore be extended to deal with *multi-task network mining* problems. Finally, acquiring labeled data for network analysis is an important but understudied problem. Although most networks already contain some label information (e.g., the partial links in a network can be used to train a link prediction model), the labels are usually incomplete and noisy. This raises an interesting question whether it is possible to design an approach for acquiring additional labeled data from alternative sources to enhance performance of network learning algorithms. These are the key issues investigated in this thesis.

1.1 Thesis Statement

This thesis focuses on multi-source and multi-task network mining problems. The techniques developed in this thesis are based on the following two conjectures. First, augmenting data from alternative sources (e.g., from crowdsourcing and other related networks) is expected to improve the performance of network mining algorithms. Second, given the inherent interdependencies between the link structure and node attributes, network learning tasks such as link prediction, community detection, and node classification are mutually related, which makes multi-task learning a natural fit for analyzing network data.

1.2 Why Multi-Source Network Mining?

The need for multi-source network mining has grown in recent years due to the following reasons. First, network data from the target domain alone may not be sufficient to yield high-quality results. In particular, any imperfections in the network data such as noisy links or missing node attribute values can have an adverse effect on the performance of the network learning algorithm. Second, network data has become more diverse. For example, it has become increasingly common for individuals to create user profiles on multiple social media Web sites. According to a recent report by the Pew Internet and American Life Project, about 51% of social network users have two or more online profiles. Each of the social media sites often maintains different aspects of information about the users. For example, Twitter provides information about user opinions and other personal communications while FourSquare [1] contains a trove of user mobility information. The users may also have a LinkedIn [2] profile containing information about their professional networks. Many of these Web sites provide easy-to-use application programming interface (APIs) to facilitate access to their public data. Similarly, advances in high throughput genomic technologies have enabled the modeling of complex biological systems using diverse types of networks, including protein interaction networks, metabolic networks, gene interaction networks, or gene regulatory networks. There has also been increasing interest in analyzing climate networks, which can be constructed using climate variables such as temperature, precipitation, and atmospheric pressure measured at different heights.

Despite the availability of the diverse sources of network data, there are very few studies that consider integrating the diverse sources of network data to improve the analysis of

[1] www.foursquare.com

[2] www.linkedin.com

complex networks. This thesis aims to develop new learning formulations that can effectively harness network data from multiple sources.

1.3 Why Multi-Task Network Mining?

A network consists of an inter-connected set of nodes, whose properties are represented by the node attributes. There is often a tight coupling between the link structure of a network and some of the attributes of the nodes. For example, individuals are more likely to befriend others who work in the same organization or attend the same school compared to those who work or attend different schools. Due to their inter-dependent relationships, many of the network learning tasks are mutually related to each other. For example, consider the product recommendation problem at an online retail store such as Amazon. Here, a "link" can be established between customers if they had bought similar products in the past. The product recommendation problem can be cast as a link prediction task on the user-product bipartite graph. It would be natural for the recommendation algorithm to first identify segments of customers who share similar purchasing behavior before making its recommendations. Identifying different customer segments can be modeled as a clustering or community detection problem. There is a close relationship between the network link prediction and community detection tasks. A good link prediction algorithm should take into consideration the community structure present in the network. Conversely, the link prediction algorithm could be used as a pre-processing step to enhance the within community links before applying a community detection algorithm [34]. This example illustrates the need to design network learning algorithms that can simultaneously solve a collection of learning tasks on one or more networks, instead of performing only a single task.

A key challenge in performing multi-task network learning is in designing a joint objective function that simultaneously performs all the related learning tasks. The shared objective function should be designed in such a way that each learning task is aided by the partial solutions obtained from other related learning tasks and the partial solutions must be intelligently combined in order to attain an optimal solution. Despite its promise, to the best of our knowledge, there has been no significant research work on multi-task network learning.

1.4 Label Acquisition for Network Mining

Adequate labeled data is the key requirement for training supervised learning algorithms for link prediction or node classification tasks. Even unsupervised learning tasks such as community detection and anomaly detection can benefit from utilizing partially labeled data (via the semi-supervised learning paradigm). In both cases, domain experts are often needed to manually peruse the data and categorize them into different labels. This is both a tedious and time consuming process, and may not always generate enough labeled data for the effective mining of large-scale networks. Annotating network data is cumbersome as the label information depends not only on the attributes of a node, but also, on the attributes and labels of its neighboring nodes. Labeling a non-network data instance is easier as its label does not depend on the label assignment for other instances. In fact, there is a subset of labeling problems known as human intelligence tasks (HITs), where it is easy (or cheap) to train non-experts to provide their reasonably accurate labels on the given data. Typical examples of HITs include digit, letter, or text recognition, image classification, and object identification in image, and video streams. It would be useful to develop a framework for exploiting labeled data (especially from HITs or other non-related domains) in order to

generate additional labeled examples for the network mining problem.

1.5 Thesis Contributions

As mentioned earlier, the focus of the thesis is in developing network mining algorithms that can exploit data from multiple sources. This is akin to network learning with side-information [57, 62]. However, previous studies on learning with side information (also known as semi-supervised learning) have focused primarily on independent and identically distributed (i.i.d.) data or network data obtained from a single source. These approaches assume there is a target data for which a learning algorithm is designed to solve with auxiliary information provided by the alternative sources (e.g., in the form of must-link and cannot-link constraints for clustering). In contrast, our approach for learning from multiple networks assumes each network is equally important and has its own learning task to be solved. In fact, the networks may have different learning parameters (e.g., number of communities in the different networks may not be the same) or learning tasks (node classification on one network and link prediction in another).

The main contributions of this thesis are as follows. In Chapter 3, we provide a matrix factorization based framework to perform joint community detection across multiple related networks, [63, 64]. In Chapter 4, we further extend the above framework to perform multi-task learning where we simultaneously perform clustering and classification on the different networks, [21, 65]. We found that the matrix factorization approach has a significant advantage over graph partitioning methods such as normalized cut especially when combining networks with different link densities. Another advantage of the framework is that it can systematically combine both content and link information from the multiple related

networks. Finally, prior information about the relationships between communities in the different networks can also be incorporated into the framework.

In Chapter 5, we present a framework to perform joint learning for link prediction and community detection [22]. Here, we have designed a novel cost-sensitive loss function that addresses both class skewness and degree skewness problems that are prevalent in most link prediction tasks. With a proper choice of the cost parameters, the proposed loss function can be theoretically shown to be equivalent to the well-known modularity measure used in community detection. We have employed a divide and conquer scheme for constructing the model, where the learning algorithm is initially applied to smaller partitions of the given network and the results obtained from each partition are systematically combined using the boosting framework.

In Chapter 6, we consider the problem of acquiring additional labeled data for supervised network mining tasks such as link prediction and node classification using crowdsourcing technology. Due to the difficulty in designing a HIT that can be easily solved, even by non-experts, we present a generic framework that transforms the network data into an image set, thereby giving a distinct visual representation of the network data for non-experts to label. Such an approach is shown to produce reliable labeled data that can be augmented to boost the performance of network mining algorithms.

Chapter 2

Background and Related Work

The study of complex networks has leaped to the forefront of data mining research spurred by the rapid proliferation of relational data generated from various physical, biological, and socio-information systems. Substantial progress has been made over the past decade to address fundamental questions such as: How are links established in a network? How do communities formed and sustained over time? What are the most influential nodes in the network? and How to infer missing attributes or links in a network? To provide answers to these and many other questions, innovative computational solutions have been developed to mine the rapidly growing repositories of network data. In this chapter we present a detailed review of advances in mining network data.

Ever since Euler first applied graph-theoretic principles to solve the *Königsberg Bridge Problem* [41], the study of networks has become increasingly popular, from the analysis of social and biological systems to the modeling of disease outbreaks, supply chains, power grids, and transportation networks. A network is typically represented as a graph of interconnected nodes, where each node represents an entity that is characterized by a set of attributes. Formally, let $\mathcal{N} = (V, E, X)$ be a network, where V is the vertex (node) set, $E \subseteq V \times V$ is the edge (link) set, and X is a matrix of nodal attributes.

2.1 Categorization of Network Types

Networks can be characterized based on the types of nodes and links they contain. A network is *homogeneous* if all the nodes in the network are of the same type. Otherwise, it is called a *heterogeneous* network. Furthermore, a network is mono-relational or *uniplex* if all the links are of the same type and multi-relational or *multiplex* if it contains links of different types [68]. For example, a co-authorship network is a uniplex network since each link indicates a pair of authors have written an article together. An online social network can often be treated as a multiplex network because two people who are linked together could be friends, relatives, fans (followers), or even complete strangers. Networks can also be characterized as *static* or *dynamic*, depending on the temporal properties of its link structure. The former is represented by a graph with fixed connectivity whereas the latter is represented by a sequence of network snapshots, each indexed by its corresponding time stamp, i.e., $\mathcal{N}_T = \{(V_t, E_t, X_t)\}_{t=1}^{T}$.

In this thesis we present algorithms for mining multiple networks. Here we assume the availability of a collection of homogeneous networks, $\{\mathcal{N}_1, \mathcal{N}_2, \cdots, \mathcal{N}_k\}$, each of which is obtained from a different data source. As used herein, the term "data source" broadly refers to any repository that houses the network data or a specific approach used to generate the data for constructing the network. For example, there are many data sources available for studying online social networks, including Facebook, Twitter, Digg, and YouTube, each of which has its own application programming interface (API) to facilitate searching and downloading data in a standardized format such as XML or JSON. Furthermore, the nodes in different networks may be connected by a set of inter-network links. In this thesis, we refer to the collection of such multiple networks along with their links as a **multi-network**.

Definition 1 *A multi-network $\mathcal{N}$ is a collection of of homogeneous networks $\{\mathcal{N}_1, \mathcal{N}_2, \cdots, \mathcal{N}_k\}$, where each homogeneous network $\mathcal{N}_i$ is an attributed graph (V_i, E_i, X_i) and the different networks are connected by a set of inter-network links, $\hat{\mathcal{E}} = \{(v_p, v_q) \mid v_p \in V_i,\ v_q \in V_j,\ i \neq j\}$.*

With this definition, it is easy to verify that multiplex networks and heterogeneous k-partite graphs are special cases of multi-networks. For example, a multi-network with homogeneous nodes and no inter-network links (i.e., $\forall i:\ V_i = V$ and $\hat{\mathcal{E}} = \varnothing$) is equivalent to a multiplex network whereas a multi-network with disjoint node sets ($\forall i, j : V_i \cap V_j = \emptyset$) and no within-network links (i.e., $\forall i : E_i = \varnothing$ but $\hat{\mathcal{E}} \neq \varnothing$) is a heterogeneous k-partite graph.

2.2 Learning Tasks on Networks

Network mining research can be broadly divided into the following tasks—network generation and characterization, link prediction, node classification, community finding, and rank analysis. We briefly review these tasks below. A more detailed exposition can be found in the following books and survey articles [35, 87, 90].

Link Prediction: Link prediction attempts to uncover previously unknown relationships in a network. It can be used, for example, to identify covert ties in an adversarial network or to predict regulatory interactions in a biological network. There are two common approaches to solve the link prediction problem: (1) by using a generative modeling approach [70, 77, 98, 103] to learn a joint probability model of the network components (node attributes and link structure) and marginalize the distribution to make a prediction, or (2) by using a discriminative approach to learn a target function that directly maps an input pair to its corresponding class [42, 47, 55]. Discriminative approaches are often preferred for several reasons. First, generative approaches require specifications of the dependence

relationships among the network components via a graphical model and assumptions about the parametric forms of the probability distributions. Second, estimation of the model parameters can be very expensive, thus requiring approximation techniques such as Markov Chain Monte Carlo (MCMC) and variational methods [27].

Node Classification: The node classification problem attempts to assign a class label to all the nodes in a network using the partial class information available on a few selected nodes in the network. This is similar to conventional pattern classification problem with the difference being that the objects to be classified are not independent and identically distributed (i.i.d.). Instead, the label of each node is dependent on the labels of its neighboring nodes. Due to such dependency, any node classification method should take into consideration the link structure as well as node attribute values when assigning class labels to the nodes. It has been previously shown that augmenting the nodal attributes of the neighbors may yield poor classification performance [14]. However, if the labels of the neighboring nodes are incorporated into the feature vector, this will improve the overall classification accuracy [14, 61]. Another important consideration for node classification is the presence of different types of regularities in the networks [110]. Here, nodes of a similar class label tend to form more links among themselves than with the rest of the network. Researchers in the past have also used probabilistic models [96] and matrix factorization [86] methods to perform the node classification task.

Rank analysis: Ranking is the process of assigning an ordering among the nodes in a network. The rank of a node reflects the measurement of some particular structural property of the network, which conveys a semantic meaning such as importance, popularity, authority, etc. As an end in itself, rankings can also be used to look for well-connected or *central* nodes in a network. In network mining, ranking is often performed using *centrality*

measures [103] such as degree, closeness, and betweenness. A popular ranking method for large directed networks such as the World Wide Web is the PageRank algorithm [75], which employs a random walk with restart approach to compute the dominant eigenvector of a matrix. Other eigenvector-based algorithms, similar to PageRank, include HITS [50] and SALSA [53]. Recently, there has been considerable interest in assigning rank values to nodes based on their community belongingness. Guimera et al. [40] introduced a metric called participation coefficient, which measures to what degree a node participates in other communities. Scripps et al. [84] introduced an alternative metric called *rawComm* for assigning ranks and roles to nodes without applying a community finding algoruthm.

Community Finding: Identifying communities in a network is closely related to the data clustering problem. Traditional clustering methods seek to find groups of objects with highly similar attribute values. In contrast, the technique of *community finding* places nodes in a network into cohesive subgroups in such a way that the nodes within a group are highly connected to each other and disconnected from nodes in other groups [63, 108]. Community finding is an ill-posed problem; there is no agreed-upon metric for evaluation. A popular graph-based metric from Newman and Girvan [71], called modularity, is based on the fraction of links within a community to those between communities. Some community finding methods do not try to completely cluster the entire network. Instead they form communities from a given seed set of nodes [30, 36]. More recently, progress in community finding has focused along finding communities in dynamic networks, where the nodes, links, and attributes change over time. Backstrom et al. [5] studied how the structural features of communities affect how nodes join and leave communities. Tantipathananadth et al. [95] proposed a new framework for tracking community changes in dynamic networks by modeling it as a graph coloring problem. Communities are then identified by approximately solving a

combinatorial optimization problem using dynamic programming.

2.3 Multi-Network Mining

As mentioned in Section 2.1, a multi-network consists of a collection of networks that are inter-connected with each other. Previous works have focused mostly on mining special cases of such networks, either as multiplex networks or heterogeneous k-partite graphs. For example, a multiplex network of articles (or documents) can be constructed by defining different types of links between the articles (e.g., based on their co-citations, similarity of keywords in abstracts or titles, and similarity of authors). There have been recent efforts to identify communities in multiplex networks [12]. Zhou et al. [115] has used the different link types to create similarity matrices between nodes which are then used to cluster the multiplex network. Researchers have also customized the well known matrix factorization techniques to cluster the multiplex networks. For example, Tang et al. [94] has proposed a linked matrix factorization approach for fusing information from multiple graphs. Lin et al. [56] also investigated a similar problem using a relational hypergraph factorization approach to detect communities of users based on various social contexts and interactions.

Another type of multi-network is a heterogeneous k-partite network, where links exist only between nodes of different types. This type of graphs has been successfully used to model relationships such as documents-words, products-users, blogs-bloggers, etc. Clustering bipartite and k-partite graphs are often referred to as co-clustering or multi-way clustering in the literature. Long et al. [58] investigated the problem of co-clustering as a matrix factorization problem and derived multiplicative update formulas for identifying the clusters. Dhillon et al. [25] presented a framework for co-clustering that minimizes the loss in mutual

information between the original joint distribution of related data and the corresponding joint distribution of the clustered data. Long et al. [59, 60] also provided a unified framework for attributes-based clustering, semi-supervised clustering, co-clustering, and graph clustering using a probabilistic framework.

The focus of this thesis is on mining multi-network data that can be conceived as a fusion of multiplex network and a heterogeneous k-partite network. Research on mining such type of networks has flourished recently. Narayanan et al. [69] propose simultaneous clustering of multiple networks as a framework to integrate large-scale datasets on the interactions among and activities of cellular components. They present an algorithm called JointCluster that finds sets of genes that cluster well in multiple networks of interest, such as co-expression networks summarizing correlations among the expression profiles of genes and physical networks describing protein-protein and protein-DNA interactions among genes or gene-products. Chen et al. [16] recently presented a co-classification approach for detecting Web spam and spammers in social media applications. They formalized the joint detection tasks as a constraint optimization problem, in which the relationships between users and their submitted Web content are represented as constraints in the form graph regularization. A pair of classifiers for detecting Web spam (url nodes) and spammers (user nodes) is simultaneously trained taking into consideration the url-url and user-user links. They demonstrated that the co-classification strategy is more effective than training the pair of classifiers independently.

2.4 Multi-task Learning on Networks

Multi-task learning is a machine learning paradigm that simultaneously learns a collection of related problems using a shared problem representation. Such an approach is desirable when the solution to one of the learning tasks can be used in learning other related tasks. It leads to a better model for each learning task, because it allows the learner to harness the commonality between the learning tasks. The merits of multi task learning over single-task learning has been discussed in detail by Caruana [13]. As mentioned in [13], if the tasks can share what they learn, the learner may find it easier to learn them together than in isolation. Thus, if we simultaneously train a classifier to recognize object outlines, shapes, edges, regions, subregions, textures, reflections, highlights, shadows, text, orientation, size, distance, etc., it may learn better to recognize complex objects in the real world, compared to learning them independently.

Multi-task learning can be considered a form of transfer learning [76]. Given a source domain $\mathcal{D}_S$ and learning task T_S, a target domain $\mathcal{D}_T$ and learning task T_T , transfer learning aims to help improve the learning of the target predictive function $f_T(.)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and T_S, where $\mathcal{D}_S \neq \mathcal{D}_T$,or $T_S \neq T_T$. Typically, in a multi-task learning the domains are identical ($\mathcal{D}_S = \mathcal{D}_T$) but the learning tasks are different. One such example is in identifying the hair color and eye color of a given face image. These are two separate tasks applied to the same data (image corpus). Most of the previous works on multi-task learning have focused on learning multiple, related classification tasks [28, 74, 106]. There has been some recent efforts to extend the formulation to learning different tasks such as classification and regression [1, 111] or regression and ranking [88]. However, none of these previous works are designed for relational data, which is the focus of this thesis.

Definition 1 (Multi-task Multi-network Learning) *Given a multi-network $\mathcal{N}$, the multi-task multi-network learning problem is to solve k learning tasks, where a learning task is associated with a subnetwork $\mathcal{N}_i = (V_i, E_i, X_i)$ of $\mathcal{N}$.*

The learning tasks stated in the preceding definition may correspond to the same class of learning problem or different classes of problems (e.g., classification, community detection, or link prediction). For example, one might be interested in classifying the nodes in one network while detecting communities in another. Furthermore, we only consider learning tasks that are mutually related; i.e., the classes or communities in one network are dependent on the classes or communities in another.

2.5 Summary

This section reviews the previous works on network mining. The past research has mostly focused on learning from a single network, temporal networks, k-partite graphs or multiplex networks. There has been very little work on combining multiple related networks nor multi-task learning on on multi-networks, which is the focus of this thesis.

Chapter 3

Joint Community Detection Across Multiple Networks

Community detection in networks is an algorithmic approach to partition the nodes in a network into cohesive groups (known as *communities*) in such a way that the nodes within a group are highly related (connected) with each other and are mostly unrelated (disconnected) from nodes in other groups. Most of the previous work has focused on finding communities in a single network or in a bipartite graph formed on heterogeneous nodes (e.g., co-clustering of articles and authors in a bibliographic network) [73, 83, 109, 112]. This chapter investigates the problem of combining link information from more than one network to improve the efficiency of community detection task.

Our work on clustering multiple networks of heterogenous nodes is motivated by its many potential applications. For example, it can be used to simultaneously find clusters of scientific papers and clusters of authors working in the same research areas. Similarly, it can also be used to perform joint clustering on Wikipedia articles and editors of the Wikipedia pages. The multiple networks may also represent relational data from different domains. For example, one could perform joint clustering of Wikipedia editors and *Digg*[1] users, where the links between Wikipedia editors and Digg users are established based on the content

[1]Digg.com is a social networking web site that allows users to share news stories with other users.

similarity between the edited Wikipedia pages and the submitted news stories in Digg. The advantages of multi-network clustering are that

- Attribute set of nodes in individual networks may not be rich enough for the purpose of clustering.
- An individual network may have noisy or partially observed links. In such a case the link structure may be enhanced by considering information from other associated networks.

A naive approach for multi-network clustering is to partition each network separately. Such an approach is useful when the link structure and subgroup information in different networks are independent of each other. However if the networks are related to each other, then the link structures in the individual networks are not only characteristic of the respective networks but often contains implicit information about the underlying clusters of other related networks. In such a scenario, we expect a joint clustering would enhance the performance of the clustering algorithm.

The proposed framework is equally applicable to clustering multiple networks created from heterogeneous nodes of the same source (e.g., Wikipedia articles and editors) or nodes from different sources (e.g., Wikipedia editors and Digg users) as long as the corresponding links between nodes in different networks can be established. One possible motivation for jointly identifying communities across different network domains is that it allows us to compare the characteristics of the similar communities present across different domains. For example, the political science community in Wikipedia may consists of lot of academic professionals and university students where as a similar community in digg.com may consist of tech savy netizens with political interests. Another motivation is that the joint clustering

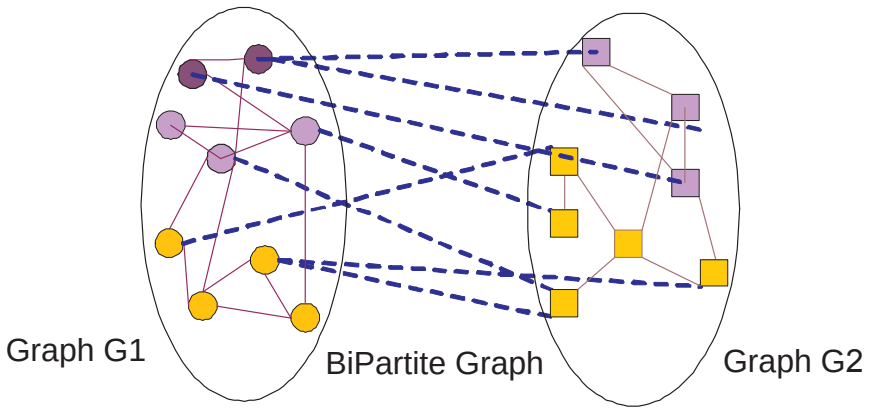

Figure 3.1 Multi Network Clustering: Graph $G_1 = (V_1, E_1)$, graph $G_2 = (V_2, E_2)$ and the bipartite graph capturing the relationship between the nodes of the networks G_1 and G_2. For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this dissertation.

allows us to incorporate the auxiliary information from multiple data sources that are related to a given network.

3.1 Preliminaries

In this thesis, we formulate the multi-network clustering problem on a pair of related networks. Our formulation can be extended to more than two networks. Let $\mathcal{G}_1(V_1, E_1)$ and $\mathcal{G}_2(V_2, E_2)$ be a pair of graphs associated with two networks. The objective of multi-network clustering is to create sets of partitions $\{\mathcal{P}_{1j}\}_{j=1}^{k_1}$ and $\{\mathcal{P}_{2j}\}_{j=1}^{k_2}$ such that $V_1 = \bigcup_{j=1}^{k_1} \mathcal{P}_{1j}$ and $V_2 = \bigcup_{j=1}^{k_2} \mathcal{P}_{2j}$. We seek a pair of functions $g_1 : V_1 \rightarrow [0,1]^{k_1}$ and $g_2 : V_2 \rightarrow [0,1]^{k_2}$ such that $g_i(v_j) = (c_1, c_2, ..., c_{k_i})$, where each $c_j \in [0,1]$ is the degree of membership node v_j belongs to cluster partition $\mathcal{P}_{ij}$. Figure 3.1 depicts the different graphs and their relationships.

In the independent clustering approach, the cluster membership functions g_1 and g_2 are learnt separately using their corresponding adjacency matrices. To simplify the notation, let A be the adjacency matrix associated with graph $\mathcal{G}_1$, where $A_{ij} = 1$ if $(v_i, v_j) \in E_1$.

Similarly, B is the adjacency matrix for graph $\mathcal{G}_2$, where $B_{ij} = 1$ if $(v_i, v_j) \in E_2$. In addition, we assume there is a third set of edges $E_3 \subseteq V_1 \times V_2$ connecting the nodes between $\mathcal{G}_1$ and $\mathcal{G}_2$. We denote the adjacency matrix for these edges as C, i.e., $C_{ij} = 1$ if $(v_i, v_j) \in E_3$.

In this study, the cluster partitions are obtained by decomposing the adjacency matrix representation of a graph into a product of its latent factors. In particular, we seek to minimize the distance function $D(A\|B)$ between the adjacency matrix A and the product of latent factors B, where:

$$D(A\|B) = \sum_{ij} A_{ij} \log \left(\frac{A_{ij}}{B_{ij}} \right) - A_{ij} + B_{ij} \tag{3.1}$$

Note that if $\sum_{ij} A_{ij} = \sum_{ij} B_{ij} = 1$, the distance function reduces to Kullback-Leibler divergence measure.

Depending on the application domain, the adjacency matrix C is either readily available as part of the data or needs to be estimated from the data. For example, consider the document-document network (A) and author-author network (B). Both networks are naturally linked by a document-author bipartite graph C. In another example, consider two networks constructed from Wikipedia editors and Digg users. Suppose we want to simultaneously cluster these two user networks such that each cluster represent users with interest in certain topics like science, sports, entertainment, etc. Here there is no natural link matrix C that is readily available to perform joint clustering. Nevertheless, it can be estimated from the data. Two users from different networks are linked if the Wikipedia pages one of them have edited has high similarity value to the news stories submitted by the Digg user.

3.2 Joint Clustering Framework

This section outlines our proposed framework for identifying cohesive subgroups in multiple networks. Our framework uses the non-negative matrix factorization technique given in [26, 52]. Let $A \in \mathcal{R}_+^{n \times n}$ and $B \in \mathcal{R}_+^{m \times m}$ be the adjacency matrices of the graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively, whereas $C \in \mathcal{R}_+^{n \times m}$ be the adjacency matrix for the links between nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$. Here $\mathcal{R}_+$ represents set of non negative real numbers. Note that our framework is applicable to both directed and undirected graphs.

3.2.1 Joint Clustering of Multiple Networks

To simultaneously cluster the networks, we minimize the following objective function with respect to X, Y, U, V and W.

$$\mathcal{J} = D(A \parallel XUX^T) + D(B \parallel YWY^T) + D(C \parallel XVY^T) \tag{3.2}$$

where $X \in \Re_+^{N \times k_1}$ and $Y \in \Re_+^{M \times k_2}$ are the corresponding cluster membership matrices for the two networks. The decomposition of A into a 3-factor XUX^t instead of a 2-factor XX^t enables the framework to deal with directed links [26, 117]. For each node i in graph G_1 , X_{ij} indicates the cluster membership of node i to cluster j. The cluster membership values are not necessarily probabilities. Large values of X_{ij} indicates greater affinity for the node i to be a member of the cluster j. Similarly, we can interpret Y_{ij}. The matrix V reflects the correspondence between the subgroups derived from the two networks.

The objective function can be written as follows:

$$\begin{aligned}\mathcal{J} &= \min_{XYUVW} \sum_{ij} A_{ij} \log \frac{A_{ij}}{[XUX^T]_{ij}} - A_{ij} + [XUX^T]_{ij} \\ &+ \sum_{ik} C_{ik} \log \frac{C_{ik}}{[XVY^T]_{ik}} - C_{ik} + [XVY^T]_{ik} \\ &+ \sum_{ks} B_{ks} \log \frac{B_{ks}}{[YWY^T]_{ks}} - B_{ks} + [YWY^T]_{ks} \end{aligned} \tag{3.3}$$

Taking the partial derivatives of $\mathcal{J}$ with respect to X and Y yield the following(the partial derivatives with respect to U, V and W can be similarly derived)

$$\begin{aligned}\frac{\partial \mathcal{J}}{\partial X_{ij}} &= \sum_{a=1}^{N} \left[\frac{-A_{ia}[XU^T]_{aj}}{[XUX^T]_{ia}} + [XU^T]_{aj} - \frac{A_{ai}[XU]_{aj}}{[XUX^T]_{ai}} \right. \\ &+ \left. [XU]_{aj} \right] + \sum_{a=1}^{M} \left[\frac{-C_{ia}[YV^T]_{aj}}{[XVY^T]_{ia}} + [YV^T]_{aj} \right] \\ \frac{\partial \mathcal{J}}{\partial Y_{ij}} &= \sum_{a=1}^{M} \left[\frac{-B_{ia}[YW^T]_{aj}}{[YWY^T]_{ia}} + [YW^T]_{aj} - \frac{B_{ai}[YW]_{aj}}{[YWY^T]_{ai}} \right. \\ &+ \left. [YW]_{aj} \right] + \sum_{a=1}^{N} \left[\frac{-C_{ai}[XV]_{aj}}{[XVY^T]_{ai}} + [XV]_{aj} \right] \end{aligned} \tag{3.4}$$

Given the objective function (3.3) and its partial derivatives, one can solve for X, Y, U, V and W using a gradient descent approach. Here, we give a converging iterative matrix factorization based update formulas for the unknown factors:

$$X_{ij} = X_{ij} \frac{\sum_{a=1}^{N} (\frac{A_{ia}[XU^T]_{aj}}{[XUX^T]_{ia}} + \frac{A_{ai}[XU]_{aj}}{[XUX^T]_{ai}}) + \sum_{a=1}^{M} \frac{C_{ia}[YV^T]_{aj}}{[XVY^T]_{ia}}}{(\sum_{a=1}^{N} [XU + XU^T]_{aj} + \sum_{a=1}^{M} [YV^T]_{aj})} \tag{3.5}$$

$$Y_{ij} = Y_{ij} \frac{\sum_{a=1}^{M} (\frac{B_{ia}[YW^T]_{aj}}{[YWY^T]_{ia}} + \frac{B_{ai}[YW]_{aj}}{[YWY^T]_{ai}}) + \sum_{a=1}^{N} \frac{C_{ai}[XV]_{aj}}{[XVY^T]_{ai}}}{(\sum_{a=1}^{M} [YW + YW^T]_{aj} + \sum_{a=1}^{N} [XV]_{aj})} \tag{3.6}$$

$$V_{ij} = V_{ij} \left[\frac{\sum_{a=1}^{N} \sum_{b=1}^{M} \frac{C_{ab}}{[XVY^T]_{ab}} X_{ai} Y_{bj}}{\sum_{a=1}^{N} \sum_{b=1}^{M} X_{ai} Y_{bj}} \right] \tag{3.7}$$

$$U_{ij} = U_{ij} \left[\frac{\sum_{a=1}^{N} \sum_{b=1}^{N} \frac{A_{ab}}{[XUX^T]_{ab}} X_{ai} X_{bj}}{\sum_{a=1}^{N} \sum_{b=1}^{N} X_{ai} X_{bj}} \right] \tag{3.8}$$

$$W_{ij} = W_{ij} \left[\frac{\sum_{a=1}^{M} \sum_{b=1}^{M} \frac{B_{ab}}{[YWY^T]_{ab}} Y_{ai} Y_{bj}}{\sum_{a=1}^{M} \sum_{b=1}^{M} Y_{ai} Y_{bj}} \right] \tag{3.9}$$

Theorem 1 *For a fixed Y, U V and W the update formula for X, given in Equation* (3.5) *monotonically decreases the objective function of the objective function defined in* (3.3).

The proof of theorem is given in the Appendix.

3.2.2 Incorporating Prior Information

Many times, we may have additional information about the correspondence between clusters in multiple networks. In what follows we give a motivation to incorporate this prior information into the objective function.

Example 1 *Consider a citation network between research articles and a co-authorship network between researchers. Suppose the articles are grouped into the following topics:* ***Algorithms, Artificial Intelligence, Databases, Cell Biology****, and* ***Genetics****. Typically,*

an author may work on multiple related topics, therefore the article partitions are not reflected as it is in the author network, rather they are further grouped into coarser clusters, namely, ***Computer Science*** *and* ***Biotechnology****. The author cluster (****Computer Science****) is related to the first three article clusters, while* ***Biotechnology*** *is related to* ***Cell Biology*** *and* ***Genetics****. We expect such prior information will enhance the joint clustering results. This information can be encoded in a* 5×2 *prior matrix:*

$$P = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

where the rows are the article clusters and the columns are the author clusters.

To incorporate prior, we first need to interpret the role of the V matrix in the objective function. As mentioned earlier, V is the 'between network' cluster correspondence matrix. The elements of V matrix reflect the relationship between the clusters between the two networks. If we have the prior knowledge about the proportion P of links between the clusters in both the networks, then it can be incorporated into the objective function (3.3) as follows

$$\begin{aligned} \mathcal{J} &= D(A \parallel XUX^T) + D(B \parallel YWY^T) \\ &+ D(C \parallel X(\lambda V + (1-\lambda)P)Y^T) \end{aligned} \tag{3.10}$$

Algorithm 1 Multi-network Clustering Algorithm

Input: Matrices A, B, C, and maximum iteration T.
Output: Matrices X, Y, and V.

Initialize X^{old}, Y^{old}, U^{old}, V^{old}, and W^{old} to random matrices.
for $i = 1$ to T **do**
  update X^{new} using (3.5)
  update Y^{new} using (3.6) with X^{new}
  update V^{new} using (3.7) with X^{new} and Y^{new}
  update U^{new} using (3.8) with X^{new}
  update W^{new} using (3.9) with Y^{new}
  set $X^{old} \leftarrow X^{new}$; $Y^{old} \leftarrow Y^{new}$; $U^{old} \leftarrow U^{new}$;
    $W^{old} \leftarrow W^{new}$ and $V^{old} \leftarrow V^{new}$
end for

where λ is a parameter provided by the user. We call $V' = \lambda V + (1-\lambda)P$ as the adjusted correspondence matrix. The λ parameter in V' controls the tradeoff between fitting V' directly to the data and fitting V' to the prior matrix P. If $\lambda = 0$, then the correspondence between clusters is given by the prior matrix. If $\lambda = 1$, then the formulation reduces to the joint clustering framework given in Equation (3.3). In situations where the proportion of data scattered between the clusters in two networks is unknown, we can use a non-informative prior where P_{ij} is 0 or 1 indicating whether the i^{th} article cluster is related to j^{th} author cluster (see Example 1).

3.2.3 Computational Complexity

This section presents analysis of the computational complexity of our algorithm. To begin with, consider the clustering of a single network. The objective function for this would be $D(A \parallel XUX^T)$ and the update formula for the cluster membership matrix X is given by

$$X_{ij} = X_{ij} \frac{\sum_{a=1}^{N} \left(\frac{A_{ia}[XU^T]_{aj}}{[XUX^T]_{ia}} + \frac{A_{ai}[XU]_{aj}}{[XUX^T]_{ai}} \right)}{\left(\sum_{a=1}^{N} [XU + XU^T]_{aj} \right)} \tag{3.11}$$

Let k be the number of clusters we wish to identify. Each X_{ij} is multiplied by an update factor, that involves several matrix multiplications. The product XU and XU^T can be computed in $O(nk^2)$ while XUX^T requires an additional computation of order $O(n^2k)$. Since A is a sparse matrix, the element-by-element division between the matrices A and XUX^T can be computed in $O(|e|_A)$, where $|e|_A$ is the number of edges in matrix A. This reduces the computation of term $\frac{A}{XUX^T}$ to $O(|e|_A k)$. Note that the update factor needs to be computed only once at each iteration and then applied, element-by-element, to each X_{ij}. Thus, the overall complexity for clustering a single network into k clusters is $O(T|e|_A k))$, where T is the total number of iterations.

Next we consider the case of two networks of comparable sizes and having the same number of clusters. The joint optimization is given by Equation (3.3) and the update formula for X is given by Equation (3.5). This differs from the update formula for a single network with an additional term that represents the relation between the two networks. If both networks are of comparable link densities, then this additional term incurs additional overhead of $O(|e|_A kT)$. By similar argument, each of the five update formulas given in Algorithm 1 requires a time complexity of $O(ekT)$, where e denotes the average number of links in a network. Putting it altogether, this implies the proposed algorithm has an overall complexity of $O(5ekT)$ where T is the maximum number of iterations.

3.2.4 Semi-Supervised Learning

Before we end this section, we discuss the similarity between the proposed joint community detection problem with well known semi-supervised learning models. The paradigm of learning from multiple related networks can be modeled as semi-supervised network learning, where the additional data from related network is used as side information in order to per-

form supervised or unsupervised learning. One way to incorporate the side-information from related networks would be to add additional knowledge derived from the side-information as constraints to the original learning problem. For example, depending on the nature and quality of the side information, a set of must-link and cannot-link constraints on the node pairs can be extracted from the side information and included in the objective function [9, 10, 57, 62, 100]. A drawback here is that it requires a good heuristic to extract the must-link, cannot-link constraints which may not be readily available. Also, it may not be possible to get a feasible clustering solution that satisfy all the constraints extracted from the side information [24].

One way to avoid using heuristics to extract constraints from the side-information is to directly use all the available information and let the algorithm extract the best suitable information from the data. In context of multiple related networks, we combine all the networks to form one giant multi-network. This is accomplished by constructing a single adjacency matrix G for the entire network as follows

$$G_{ij} = \begin{cases} A_{ij} & i, j = 1, 2, ...n, \\ C_{ij} & i = 1, 2, ..n \text{ and } j = n+1, ..m \\ C_{ji} & j = n+1, ..n+m \text{ and } i = 1, ...n \\ B_{ij} & i = n+1, ..n+m \text{ and } j = n+1, ..n+m \ . \end{cases}$$

We then apply any manifold learning algorithms like label propagation or normalized cut [92] algorithms on the entire multi-network G. It should be noted that the proposed joint factorization framework can be thought of as novel approach to incorporate side information as well as prior knowledge into the learning framework for performing clustering on networks. In the next section, we present the results comparing the performance of the proposed

algorithm against its manifold learning counterpart.

3.3 Experimental Evaluation

We have evaluated the effectiveness of the proposed algorithm on both synthetic and real world network data. The details of the real world network and synthetic networks are given below.

3.3.1 Baseline Algorithms and Evaluation Metrics

As a baseline, we used three clustering algorithms. The first one is the normalized cut (`Ncut`) algorithm by [92]. The other two algorithms are part of the objective function given in (3.3). The first term in the objective function (3.3) refers to independent clustering of a single network. The first two terms corresponds to the co-clustering of a single network using the bi-partite graph between two networks. We compare our joint clustering algorithm against the independent clustering (denoted by `Ind`) of single network and co-clustering (denoted by `CoC`). We denote our proposed framework as `Joint` or `Joint + Prior` in the remainder of this section. For a fair comparison, we applied the normalized cut algorithm on the entire multi-network $\mathcal{G}$ described above (instead of $\mathcal{G}_1$ and $\mathcal{G}_2$ separately).

We use the normalized mutual information (NMI) measure to evaluate clustering results. It is defined as follows. Let $\mathcal{C} = \{C_1, C_2, ...C_k\}$ denote the true set of clusters. Let $\hat{\mathcal{C}} = \{\hat{C}_1, \hat{C}_2, ...\hat{C}_k\}$ denote the cluster obtained from the algorithm. Then mutual information between them is defined as

$$MI(\mathcal{C}, \hat{\mathcal{C}}) = \sum_{C_i, \hat{C}_j} p(C_i, \hat{C}_j) \frac{p(C_i, \hat{C}_j)}{p(C_i)p(\hat{C}_j)}$$

and the normalized mutual information is given by

$$NMI(\mathcal{C}, \hat{\mathcal{C}}) = \frac{MI(\mathcal{C}, \hat{\mathcal{C}})}{H(\mathcal{C})H(\hat{\mathcal{C}})}$$

where $H(\mathcal{C})$ and $H(\hat{\mathcal{C}})$ denote the entropies of partition $\mathcal{C}$ and $\hat{\mathcal{C}}$ respectively.

3.3.2 Synthetic Data set

Our synthetic data set is generated as follows. First, the number of nodes and number of clusters (k) in each network are given. In our experiment, the number of clusters in each network is fixed to be 4 with 400 data points in each cluster. Within each cluster i, a link is created between any two nodes with probability p_1. On the other hand, an inter-cluster link is created between a node in cluster i and nodes in other clusters in the network with probability p_2. In addition, links are also created between nodes from different networks. We create links between networks $\mathcal{G}_1$ and $\mathcal{G}_2$ with probabilities q_1 and q_2, where the former is the probability of link between corresponding clusters and the latter is the probability of noisy link between non-corresponding clusters. For example, if networks $\mathcal{G}_1$ and $\mathcal{G}_2$ have 4 clusters each, q_1 is probability of link between $cluster_i$ in network $\mathcal{G}_1$ and $cluster_i$ in network $\mathcal{G}_2$. q_2 is the probability of link between $cluster_i$ in $\mathcal{G}_1$ and $cluster_j$ in $\mathcal{G}_1$ with $i \neq j$.

3.3.3 Complexity Verification

In this section, we use the synthetic network to verify the theoritical computational complexity derived in section 3.2.3. Here we computed the time for factorization of single network with $k = 4$ clusters. We set the inter-cluster link probability to be 0.01 and intra-cluster link probability to be 0.001. We then varied the number of nodes in the network from 400 to 4400

in step size of 800 nodes. For each network we computed the factorization $D(A \parallel XUX^T)$ and plot the computation time as function of number of links in the network. This is shown in Figure 3.2. Clearly, the computation time is seen to be linear in the number of links in the network as derived in section 3.2.3.

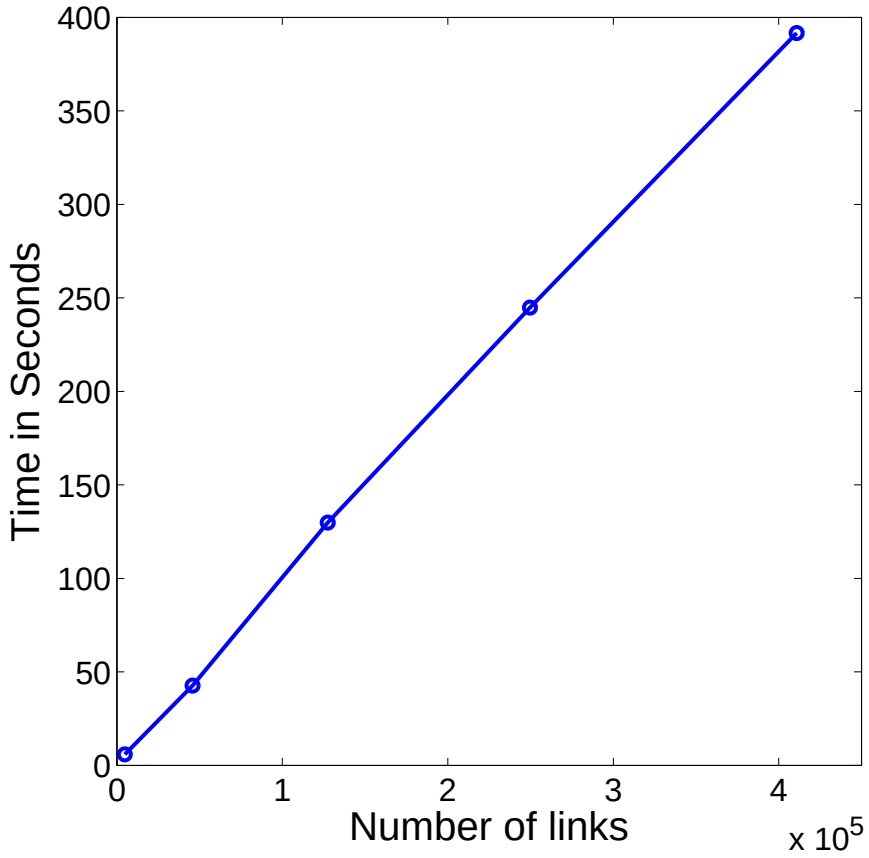

Figure 3.2 This plot shows the variation of computation time with increase in number of links in the network (due to increase in nodes).

3.3.3.1 Effect of noise in one network

We created two networks namely, Network A and Network B using the above mentioned parameters. Network A was generated with $p_1 = 0.4$ and $p_2 = 0.35$. The links between the network was generated with $q_1 = 0.6$ and $q_2 = 0.45$. Network B was generated with $p_1 = 0.5$

and we varied the inter cluster link probability for Network B from 0.1 to 0.4 in step size of 0.1. These are the noisy link in the Network B. We studied the effect of varying noisy links in Network B on the cluster NMI on network A. The results are plotted in Figure 3.3. As expected, low level of noise in network B helped in identifying the clusters in Network A with higher accuracy. As the noise level increases in Network B, the NMI of clusters obtained in Network A decreases. This reflects our belief that if one of the networks is less noisy then it helps in improving the clustering accuracy of the other network. `Ncut-M` and `Ncut-I` in the figures 3.3 and 3.4, refers to normalized cut algorithm applied on the Multi graph and Individual network respectively.

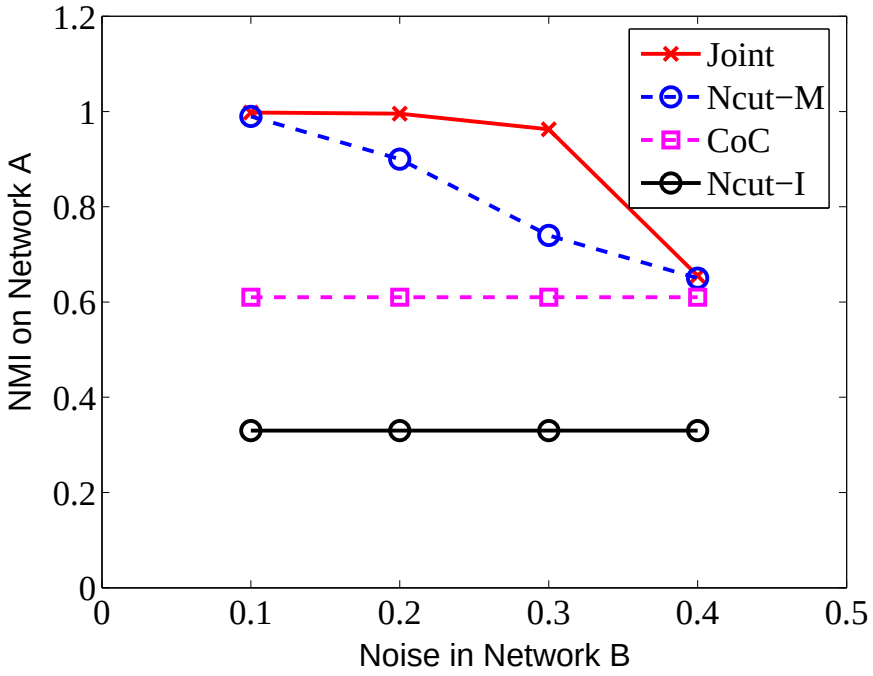

Figure 3.3 Effect of Noisy links in Network B on clustering performance of Network A ($p_1 = 0.4$ and $p_2 = 0.35$)

3.3.3.2 Effect of noise in between the networks

Here, we tested the effect of noise between the networks on the clustering accuracy. To do this we created two networks with following parameters. Network A with $p_1 = 0.4$ and $p_2 = 0.35$ and Network B with $p_1 = 0.5$ and $p_2 = 0.45$. We created the links between them with $q_1 = 0.6$. We varied the noise between the networks by varying q_2 from 0.1 to 0.5 in step size of 0.1. The results of clustering accuracy on Network A is shown in Figure 3.4. When the link between the networks are less noisy, the `CoC` algorithm gives good result. The `Joint` algorithm gives better results. This is because the information flow between the networks is more reliable and the link structure in Network B helps in improving the performance of `Joint` clustering over the `CoC` clustering.

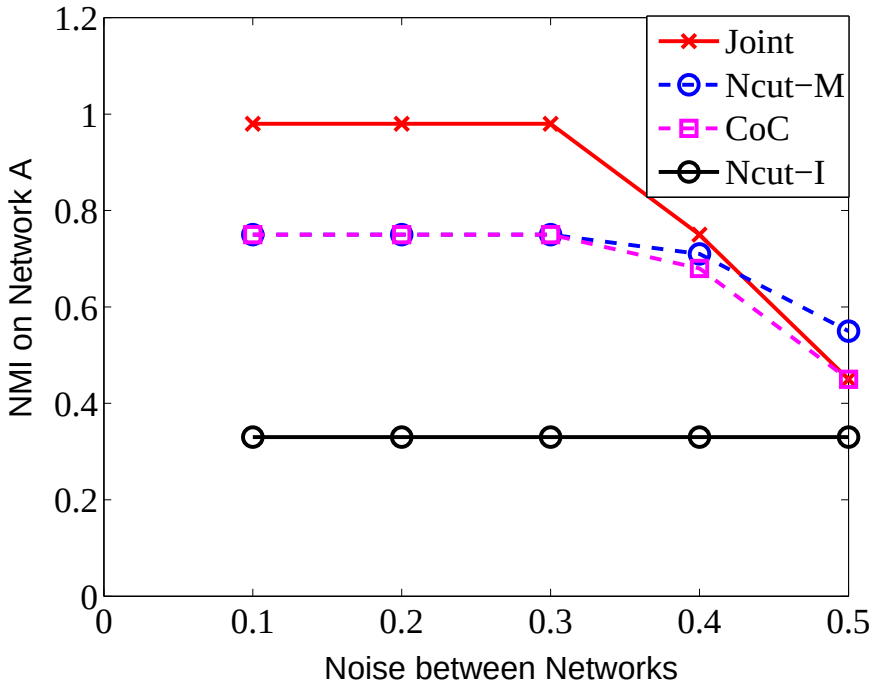

Figure 3.4 Effect of Noisy links between networks on the clustering performance of Network A ($p_1 = 0.4$ and $p_2 = 0.35$)

3.3.4 Wikipedia Dataset

We use the Wikipedia dump from Oct-09-2009 for our experiments. We have chosen four topics as the ground truth clusters—Biology, Natural Science, Computer Science and Social Science. Each of the four topics are further divided into subtopics which are shown in Table 3.1. We collected roughly 20K articles, with 5K articles in each category. After removing stubs and other smaller articles we were left with 10K articles and 53K editors (who have edited the articles). We removed articles/editors that do not have sufficient links (less than 3 links) with other articles/editors in our corpus. Our final data set contains 6403 articles and 5361 editors. A visual representation of the adjacency matrices of the article and editor networks is shown in Figure 3.5. Our goal of clustering is to identify the 12 sub-categories in the article network and relate them to 4 categories in the editor network. Such a clustering is useful in other real world application, where in, low level label information like sub topics are assigned to each article in the article corpus and high level label information are assigned to authors based on their research interests. Joint clustering can then used to identify and associate the related communities between these networks and propagate the high level labels from authors networks to more refined low level labels in the article clusters.

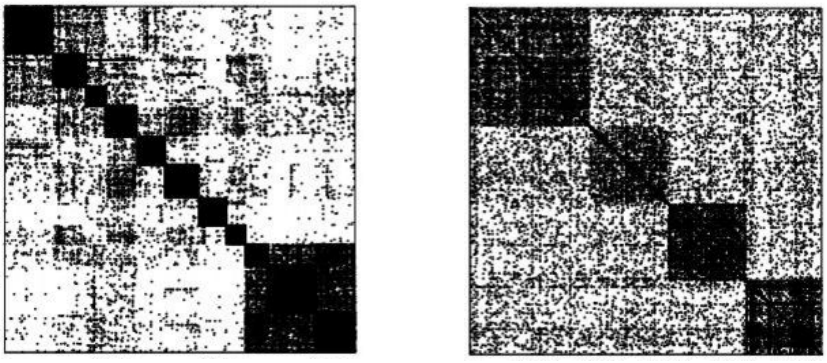

FIGURE 3.5: Spy plot for Article (Left) and User (Right) networks in Wikipedia

The Wikipedia data set is particularly challenging. Firstly, the editors do not seem to

TABLE 3.1: Data Category and Sub Category

Category User clusters	Sub-Categories Article clusters
Political Science	Civil-Rights Liberties(878); Imperialism(601); Nationalism(368);
Natural Science	Physics(568); Earth Sciences(513); Astronomy(613)
Computer Science	Algorithms(112); Operating Systems(395); Computer Architecture(350)
Biology	Zoology(392); Anatomy(897); Cell-Biology(716);

TABLE 3.2: The link distribution between different author clusters in Wikipedia data set

	Political Science	Natural Science	Computer Science	Biology
Political Science	8313	1113	749	844
Natural Science	1113	3398	657	592
Computer Science	749	657	5337	516
Biology	844	592	516	4806

have a fixed domain of interest. As seen in the spy plot in Figure 3.5, a good proportion of editors have edited articles in all the four categories. We assign a ground truth label to each user based on the category for which the user has made the most number of contributions. Secondly, although each editor has his/her own Wikipedia page, many of these pages do not contain enough useful features that can be used to identify the cluster of an editor. Thirdly, the links between articles tend to be noisy. The article-article spy plot in Figure 3.5 shows 9 visually distinct groups even though there are 12 article clusters. This is because, in our sample, the articles in some of the sub categories are highly connected to other realted sub categories. It is therefore not visually discernable.

We first clustered the article network and user network independently and used them as our baseline result. As shown in Table 3.3, independent clustering of article gives a NMI of 0.30 and co-clustering of article network using the article-editor bipartite graph increases the NMI to 0.38. However, the joint clustering gave a slightly higher NMI of 0.41, highlighting

the importance of information flowing from the other network. Joint clustering with prior information increases the NMI further to 0.45. The normalized cut algorithm produced the highest NMI of 0.55 on the article network. The confusion matrix of `Ncut` and `Joint` algorithms on article network is given in Table 3.5 and Table 3.4.

TABLE 3.3: Average Cluster NMI on Wikipedia dataset

Experiment	Article	Editor
Normalized Cut on Multi Graph (12 clusters)	0.550	-
Normalized Cut on Multi Graph (4 clusters)	-	0.010
Independent clustering	0.304	0.080
Co-clustering	0.381	0.208
Joint clustering without Prior ($\lambda = 1$)	0.405	0.213
Joint clustering with Prior ($\lambda = 0.5$)	0.454	0.259

As seen in the Table 3.4, the `Ncut` algorithm on the multi graph G has found 4 predominant clusters in the article network, one for each category. For example, clusters $1, 6, 9, 11$ in the confusion matrix of `Ncut` algorithm represent predominantly represent the underlying major four categories and rest of the columns are very sparse. However, the `Joint` clustering algorithm has found the fine sub categories in the article network. For example, the clusters $1-3$ predominantly represent the sub categories of political science, clusters $4-6$ predominantly represent natural science subcategories and clusters $10-12$ predominantly represent the subcategories of biology. The algorithm has found only two predominant subcategories (columns $7-8$)for the computer science category.

In the editor network, the `Joint + Prior` has out performed all other approaches. The `Ncut` on multi graph could not identify any of the major four categories. Both `Joint` and `CoC` algorithms has shown similar performance with NMI of 0.21. A simple `Joint` algorithm did not give any additional benefit compared to the `CoC`. However, adding the non-informative

TABLE 3.4: Confusion matrix of article network using the `Ncut` algorithms on the multi graph G gave NMI 0.55. The columns 1,6,9 and 11 are the dominant clusters identified by the `Ncut` algorithm. These four dominant clusters correspond to the four major categories and the algorithm has failed to identify the sub categories.

	1	2	3	4	5	6	7	8	9	10	11	12
Ground Truth	Clusters Output by `Ncut` Algorithm											
Civil rights	536	286	42	0	1	3	2	4	1	0	3	0
Imperialism	331	0	222	2	1	6	21	2	0	1	15	0
Nationalism	286	2	17	1	0	0	3	1	11	2	45	0
Physics	3	0	270	0	1	21	247	0	3	0	23	0
Earth Sc	17	21	16	161	234	27	26	2	8	0	1	0
Astronomy	49	0	10	0	0	347	20	179	2	1	5	0
Algo	0	0	0	0	0	1	7	0	104	0	0	0
OS	1	0	0	0	0	20	2	1	371	0	0	0
Architecture	19	1	101	0	0	2	213	0	0	0	0	14
Zoology	25	0	18	0	0	1	1	0	0	1	346	0
Anatomy	1	0	0	0	0	2	1	1	0	0	892	0
Cell Biology	5	1	11	0	0	2	0	1	5	200	491	0

TABLE 3.5: Confusion matrix of article network using the `Joint` clustering method gave NMI 0.40. Unlike the `Ncut` algorithm, the `Joint` clustering is able to identify the finer sub categories. For example, columns 1,2,3 predominantly identifies the sub categories of political science category. Similarly, columns 10,11 and 12 relates to sub categories of the biology category.

	1	2	3	4	5	6	7	8	9	10	11	12
Ground Truth	Clusters Output by `Joint` Algorithm											
Civil rights	21	416	14	1	2	265	0	2	0	112	34	11
Imperialism	208	177	110	0	15	54	1	0	0	31	4	1
Nationalism	14	261	14	0	36	7	3	7	0	18	6	2
Physics	8	1	9	4	295	245	1	0	3	0	2	0
Earth Sc	18	4	25	248	171	13	0	7	5	6	9	7
Astronomy	120	1	96	78	13	130	162	12	0	0	0	1
Algorithm	0	0	0	2	0	0	62	48	0	0	0	0
OS	2	0	0	1	0	0	122	251	0	0	12	7
Architecture	32	9	26	195	13	55	11	0	0	1	3	5
Zoology	23	13	20	0	19	38	1	2	8	4	176	88
Anatomy	0	1	0	0	47	54	11	0	11	172	393	208
Cell Biology	5	9	4	0	32	12	1	30	4	187	286	146

prior boosted the performance of the joint clustering algorithm.

3.3.5 Digg Data Set

The web site www.digg.com is a popular social bookmarking web site where each individual users bookmark URL's and share them publicly with other users. The goal is to identify different user community based on the topics of the bookmarked URL's. As mentioned in the abstract and introduction, the objective of this work is to investigate whether it is possible to combine information from several networks to improve community detection. However, one of the challenges in demonstrating this effect empirically is the lack of ground truth information about the true clusters of various domains. So, the main reason we choose Wikipedia and Digg data sets for our experiments is the availability of the ground truth class labels that can be used to evaluate the performance of our joint community detection framework. Furthermore, though there may not be a direct correspondence between Wikipedia editors and Digg users, their community structures are often defined based on the topics of the articles they have edited or posted. Since there exists common topics among articles in both networks, this information that can be harnessed to improve their community detection tasks. The idea of using Wikipedia as an auxiliary data source for improving clustering has also been investigated before (see for example, the works by [6]), though none of the previous works consider the Wikipedia as source of auxiliary network data for user community detection. Our experiment suggests that Wikipedia is a potentially useful source to improve clustering of users in a domain such as Digg.com as well. The exact data collection mechanism is described below.

We first sampled 5670 digg users who have bookmarked URLs on the following three topics: *Politics, Computer Science*, and *Natural Science*. We formed a user-user connectivity

matrix from the user-URL matrix. Two users are linked if they have at least $\tau = 5$ URLs in common. We sampled 4206 Wikipedia editors that belong to aforementioned 3 categories. We established the links between the digg users and the Wikipedia editors by first establishing a similarity between the bookmarked URLs and the edited articles. Each URL bookmarked at digg.com has a title and a short description about the content of the web site. The digg url-word matrix and Wikipedia article-word matrix are used to establish a "weighted link" between a digg url and a Wikipedia article. Specifically, the weight of the link corresponds to the cosine similarity between the words in the title and description of a URL and the words that appear in the content of a Wikipedia article. We finally establish the link between a digg user and Wikipedia editor if there is high similarity between the contents of bookmarked URL and edited article. Here again, we assign a ground truth label to each editor (user) based on the category for which the editor has made the most number of contributions. Figure 3.6 shows the adjacency spy plot for two networks. In both these networks a good proportion of editors/users have contributed articles in all three categories. However, notice that the there is large number of links between the first two user communities in the digg network. These two communities refer to the *politics* and *computer science*. It indicated that in our sample, the users who have bookmarked politics related URL have also bookmarked technology related URLs.

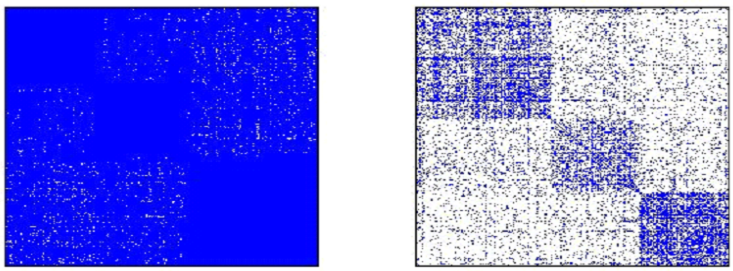

FIGURE 3.6: Adjacency matrix plot for Digg user and Wikipedia editor networks.

TABLE 3.6: Confusion matrix on digg user and wiki editor network using `Ncut` on Multigraph

`Ncut` on Digg+ Wiki MultiGraph						
Digg User Network				Wiki Editor Network		
167	1481	0		108	1708	0
997	638	0		707	492	0
1952	435	0		1113	77	1
NMI:0.185				NMI:0.307		

TABLE 3.7: Confusion matrix of Digg users and Wikipedia editors using the proposed joint clustering method.

`Joint` on Digg+ Wiki MultiGraph						
Digg User Network				Wiki Editor Network		
1242	63	343		102	70	1644
169	220	1246		631	270	298
338	1889	160		122	1023	46
NMI:0.387				NMI:0.418		

We first applied the `Ncut` algorithm on the multi graph containing both digg and wiki networks. Both the networks contain three corresponding clusters. The confusion matrix is shown in Table 3.6. Clearly, the `Ncut` algorithm identified only two clusters on each network. This is because, the digg network had predominantly two clusters. Table 3.7 gives the confusion matrix of digg and Wikipedia network using the proposed `Joint` approach. It has identified the three dominant community on each network. Since there was a natural correspondence between the three clusters on both the networks, adding prior information (identity matrix of size 3) did not give any additional improvements.

3.4 Summary

In this chapter we have discussed the problem of learning cohesive subgroups and their correspondences in multiple related networks. Our experiments reveal that the joint clustering of multiple networks gives better results in terms of normalized mutual information between

the actual clusters and the clusters identified by algorithm. We have also introduced the idea of using a prior to guide the clustering process. We have performed a through analysis of the convergence of the proposed algorithm and we have also given heuristics for faster convergence. The proposed algorithm is of order $O(n^2T)$ in complexity where n is number of nodes on either network and T is number of iterations. The scalability of the proposed algorithm for networks with millions of nodes will be pursed in future.

Chapter 4

Joint Clustering and Classification on Multiple Networks

In the previous chapter, we focused on learning a single task (community detection) simultaneously on a two related networks. In this chapter we extend the framework for multi-task learning on two related network data. Specifically, we present a novel framework that enables one to perform classification on one network and community detection in another related network. Multi-task learning is accomplished by introducing a joint objective function that must be optimized to ensure the classes in one network are consistent with the link structure, nodal attributes, as well as the communities detected in another network. Experiments performed on both real-world and synthetic data sets demonstrate the effectiveness of the joint framework compared to applying classification and community detection algorithms on each network separately.

The motivation for this study is two-fold. First, the rapid proliferation of online social and information networks raises the question whether we can leverage network data from known information sources (say, Wikipedia) to enhance the network learning tasks. In a previous chapter, we have demonstrated the advantages of combining data from multiple related networks to improve community detection. However, the approach presented in Chapter 3 considers the same learning task—community detection—on all networks. Here, we extend

the analysis to the case when a different learning task is performed on each network. For example, one might be interested in leveraging the knowledge about the classes of Wikipedia editors (based on the articles they have edited) to identify the communities of users at `Digg.com`, a social news Web site. Though the correspondence between the identities of Digg users and Wikipedia editors may not be known, their community structures and classes are potentially well-aligned as they are based on the topics of articles posted or edited by the users.

Second, by comparing the classes and communities across different networks, one could potentially explain the identified communities in one network in terms of the known classes defined in another network, especially when there is high connectivity between nodes that belong to a community and its associated class. Furthermore, it is also possible to perform a comparative network analysis to identify the classes (clusters) found in one network but not the other. For example, given the known classes of users at Facebook, can we identify groups of MySpace users whose link structure and nodal behavior do not correspond to any known groups in Facebook?

One approach to solve this multi-task multi-network learning problem is to combine all the networks into one giant graph and apply the classification or community finding algorithms to the combined graph. However such an approach has many limitations. First, by applying a single algorithm to the combined graph, we have no control over the number of clusters or classes that will be found in each of its underlying networks. In particular, as will be shown in our experiments, this approach does not work well when the number of classes in one network is different than the number of communities in another network. Furthermore, it may lead to a suboptimal solution as it attempts to fit a global model to a graph that contains local networks with their own distinctive properties.

The main contributions of this chapter are as follows. First, we extend the framework in Chapter 3, for joint classification and community detection in heterogeneous network data. The framework is applicable even when the number of classes in one network differs from the number of communities in another network. In addition, we have drawn parallels between the update formula and the label propagation algorithm [114, 118]. We have also included extensive experiments using synthetic data sets to assess the performance of our proposed framework under different multi-network parameter settings. In particular, these experiments help to shed light into fundamental questions such as (1) Can clustering on one network help to improve classification on another network, and vice-versa? (2) Does combining multiple related networks help in solving a task better than solving the same task independently on individual networks? (3) How does the presence of noisy links in different networks affect performance of each learning task?

4.1 Joint Learning Framework

The notations used in this chapter are similar to the ones used in chapter. We summarize these notions in Table 4.1.

The objective function for joint clustering and classification is very similar to the objective function (3.2) defined in Chapter 3 with an additional term for aligning the label information present in the network.

$$\begin{aligned} \mathcal{L} = \min_{X,U,V,Y,W} \quad & D(A \parallel XUX^T) + D(C \parallel XVY^T) \\ + \quad & D(B \parallel YWY^T) + D(\beta L \parallel Y_l) \end{aligned} \tag{4.1}$$

TABLE 4.1: Summary of notations used in the chapter.

Symbol	Description
$\mathcal{G}_1$	A homogeneous network for detecting communities
$\mathcal{G}_2$	A homogeneous network for classification
$\mathcal{G}_{12}$	A bipartite graph connecting nodes between $\mathcal{G}_1$ and $\mathcal{G}_2$
$v_{ij} \in V_i$	A node in network $\mathcal{G}_i$ ($i = 1$ or 2)
$\mathcal{E}_i$	The set of links in network $\mathcal{G}_i$
$\mathcal{E}_d$	The set of links in the bipartite graph $\mathcal{G}_{12}$
k_1	number of communities in $\mathcal{G}_1$
k_2	number of classes in $\mathcal{G}_2$
l	number of labeled nodes in $\mathcal{G}_2$
A	An $n \times n$ adjacency matrix for network $\mathcal{G}_1$
B	An $m \times m$ adjacency matrix for network $\mathcal{G}_2$
C	An $n \times m$ adjacency matrix for bipartite graph $\mathcal{G}_{12}$
X	An $n \times k_1$ pseudo-label matrix for community membership of the nodes in $\mathcal{G}_1$
Y	An $m \times k_2$ pseudo-label matrix for class membership of the nodes in $\mathcal{G}_2$
V	A $k_1 \times k_2$ community-class correspondence matrix
L	An $l \times k_2$ true class membership matrix for the labeled nodes in $\mathcal{G}_2$

The first term in the objective function deals with the clustering of nodes in $\mathcal{G}_1$ by factorizing the adjacency matrix A into a product involving the pseudo label matrix X. The last two terms deal with the classification of nodes in $\mathcal{G}_2$ by estimating the pseudo label matrix Y, taking into account both the link structure (B) and class information (L). Thus, Y_l is an $l \times k_2$ sub-matrix of Y consisting on rows of Y corresponding to the labeled data points. Thus, the last term in the objective function, $D(L \parallel Y_l)$, does not apply to unlabeled nodes in network $\mathcal{G}_2$. Meanwhile, the second term in the objective function is used to learn the relationship between the clusters found in network $\mathcal{G}_1$ and the classes obtained for network $\mathcal{G}_2$. The association between the clusters and classes are encoded by the cluster-class correspondence matrix V. The constant β indicates the factor by which Y_l would be fit to the ground truth label L. Since L is fixed binary label matrix, the constant factor β can be absorbed into it. That is, $L_{ij} = \beta$ if the node $v_{2i} \in V_2$ belongs to class j and zero otherwise. In what follows we would not explicitly mention the label factor β.

Here again, the optimization problem is solved using an alternating minimization scheme. The update formula for matrices X, U, W, and V are same as (3.5),(3.8), (3.9) and (3.7) respectively. The update formula for Y_{ij} depends on whether the node is labeled or not. It is given by

$$Y_{ij} = \begin{cases} Y_{ij}\frac{\sum_a(\frac{B_{ia}[YW^T]_{aj}}{[YWY^T]_{ia}}+\frac{B_{ai}[YW]_{aj}}{[YWY^T]_{ai}})+\sum_a \frac{C_{ai}[XV]_{aj}}{[XVY^T]_{ai}}}{(\sum_a [YW+YW^T]_{aj}+\sum_a [XV]_{aj})}, & i > l \ ; \\ Y_{ij}\frac{\sum_a(\frac{B_{ia}[YW^T]_{aj}}{[YWY^T]_{ia}}+\frac{B_{ai}[YW]_{aj}}{[YWY^T]_{ai}})+\sum_a \frac{C_{ai}[XV]_{aj}}{[XVY^T]_{ai}}+\frac{-L_{ij}}{Y_{ij}}}{(\sum_a [YW+YW^T]_{aj}+\sum_a [XV]_{aj})+1}, & i \leq l. \end{cases} \tag{4.2}$$

We highlight the advantages of our proposed multi-task learning technique compared to single-task learning with label propagation and cut-based graph partitioning algorithms in the next section.

4.2 Joint Learning vs Independent Learning

In this section, we will give an insight into the exact mechanism by which the proposed framework performs the iterative clustering and classification between the networks until the community structure and classes crystalize in each of the network. We explain this from the point of view of classification problem, though similar explanation can be made for the clustering problem. We also compared our joint learning approach to independent learning using the well-known label propagation (for classification) and graph partitioning (for clustering) algorithms.

4.2.1 Joint Factorization vs Label Propagation

Consider a binary classification problem in a network where each node belongs to either class 1 or class 2. Suppose the class information is encoded in a two dimensional row vector, where $[1, 0]^T$ represents a node assigned to class 1 and $[0, 1]^T$ represents a node assigned to class 2. Unlabeled nodes are assigned a vector $[0, 0]^T$. A node classification algorithm can be designed to propagate a fraction of the class information from a labeled node to its neighbors at each iteration. An unlabeled node will sum up the label vector it receives from each of its neighbor. After a sufficient number of iterations, an unlabeled node in the network whose label vector is denoted by $[l_i^1, l_i^2]$ will be assigned to class 1 if $l_i^1 > l_i^2$ and to class 2 otherwise. This is the learning strategy employed by a broad class of *label propagation* algorithms [114, 118].

Two important parameters that govern the propagation of labels between nodes in a network are (1) *propagation structure* and (2) *rate of propagation*. The former refers to the link structure that defines the neighborhood structure of each node in the network while the latter determines the fraction of amount by which the label information is propagated to neighboring nodes at each iteration. The *rate of propagation* is usually modeled as a function of the weight associated with the link between an adjacent pair of nodes. More formally, the update formula for the label propagation algorithm is given by

$$Y^{(t)} \leftarrow \alpha \mathbf{P} Y^{(t-1)} + (1 - \alpha)\mathbf{L}, \tag{4.3}$$

or equivalently,

$$Y_{ij}^{(t)} = \alpha \sum_a \mathbf{P}_{ia} Y_{aj}^{(t-1)} + (1 - \alpha) L_{ij} \tag{4.4}$$

where $\mathbf{P}$ is the propagation matrix constructed from the adjacency matrix of the network. There are several forms of label propagation matrices proposed in the literature, including $\mathbf{P} = \mathbf{D}^{-1}\mathbf{B}$ [118] and $\mathbf{P} = \mathbf{D}^{-\frac{1}{2}}\mathbf{B}\mathbf{D}^{-\frac{1}{2}}$ [114], where $\mathbf{B}$ is the adjacency matrix and $\mathbf{D}$ is a diagonal matrix who diagonal entries are $\mathbf{D}_{ii} = \sum_j \mathbf{B}_{ij}$. Next, we show that the multiplicative update formula given by our matrix factorization framework have similar mathematical form, which suggests that our multiplicative update formula can be viewed as a form of multi-task multi-network label propagation.

Consider the problem of classification on a single network $\mathcal{G}_2$ without using the information from the bipartite graph $\mathcal{G}_{12}$. The objective function involves minimizing the terms $D(B \parallel YWY^T) + D(L \parallel Y_l)$. The update formula for Y can be obtained by considering only the relevant terms in Equation (4.2). Assuming the links in the network are undirected, this imposes symmetry restriction on the matrices B and W. Furthermore, let $M_{ij} = \frac{B_{ij}}{[YWY^T]_{ij}}$, which represents a weighted adjacency matrix, whose large weights are associated with links that were incorrectly predicted by YWY^T in the previous iteration. Hence, the update formula Y for labeled nodes can be re-written as

$$\begin{aligned} Y_{ij} &= Y_{ij} \frac{\sum_a \left(\frac{B_{ia}[YW^T]_{aj}}{[YWY^T]_{ia}} + \frac{B_{ai}[YW]_{aj}}{[YWY^T]_{ai}}\right) + \frac{L_{ij}}{Y_{ij}}}{\sum_a (YW + YW^T) + 1} \\ &= Y_{ij} \frac{\sum_a M_{ia}[YW]_{aj} + \frac{L_{ij}}{Y_{ij}}}{(\sum_a [YW]_{aj}) + 1} \end{aligned} \tag{4.5}$$

Notice that the denominator term approximately normalizes the columns of the matrix YW. Let $\mathbf{D}$ be the diagonal matrix with diagonal entries consisting of the column sums of YW matrix. That is, $D_{ii} = \sum_a [YW]_{ai} + 1$. Then the column normalized label matrix is given by $\tilde{Y} = YW\mathbf{D}^{-1}$. Let Y^j represent the j^{th} column of Y (that is, the label information

of j^{th} class), then the above update formula can be written as

$$Y_{ij} \;=\; Y_{ij} \sum_{a} M_{ia} \tilde{Y}_{aj} + \frac{L_{ij}}{D_{jj}} \tag{4.6}$$

Comparing the equations (4.4) and (4.6), we notice that the multiplicative update formula has the same mathematical form as the label propagation algorithm. The key difference between them is that the multiplicative updates have a varying propagation matrix M, instead of a fixed propagation matrix $\mathbf{P}$. The role of α in (4.4), is to establish consistency between the actual and predicted labels on the labeled data set. This is done by the factor β in (4.6). Recall that β is absorbed into L, i.e $L_{ij} = \beta$ if the node $v_{2i} \in V_2$ belongs to class j and zero otherwise.

The update formula for unlabeled nodes can be similarly shown to have same mathematical form as (4.4) albeit with different propagation matrix. This aspect makes it very adaptive in that, the propagation matrix can be updated at each iteration to accommodate more information from the other network. At each iteration, the partial cluster information from network $\mathcal{G}_1$ and the partial label information from $\mathcal{G}_2$ determines the propagation structure and the propagation rate for the classification problem. The label information thus propagated would minimize the class ambiguity among the unlabeled nodes. The enriched class information is again propagated back for performing the clustering in $\mathcal{G}_1$.

4.2.2 Joint Factorization vs Graph Cuts

The earliest algorithms on identifying communities in a given network were borrowed from the graph theory literature, where, subset of the edges were removed to induce partition on

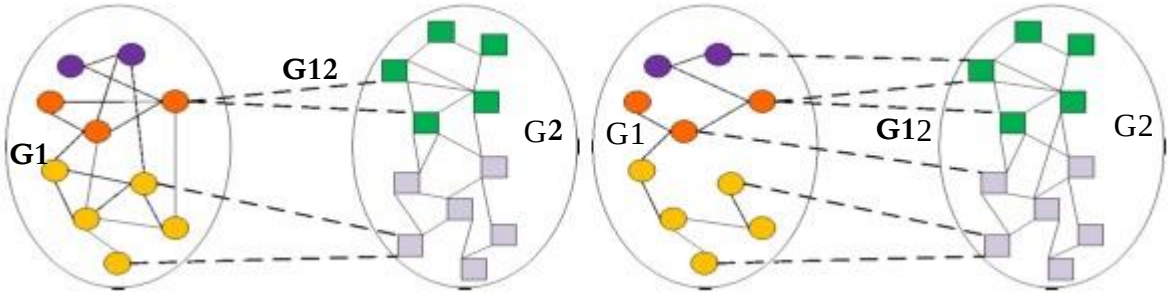

Figure 4.1 Sample multi-network to illustrate the functioning of graph cut based partitioning techniques. **Left**: The link density in the bipartite graph $\mathcal{G}_{12}$ is sparse. As as result, the graph cut based algorithm would first disintegrate the multi-network into individual networks and then continue to split individual networks based on their link densities. **Right:**: The link density is very different between network $\mathcal{G}_1$ from $\mathcal{G}_2$, as a result, one of them may get split many times while the other remain intact. Such problem wont arise in proposed joint factorization.

the vertex set. Given a graph $\mathbf{G = (V,E)}$, a cut <M,N> is a partitioning of the vertex set $\mathbf{V} = M \bigcup N$ of graph by removing edges between the vertex sets M and N. The size of the cut is defined as number of edges removed from the edge set **E**. Let A be the adjacency matrix of the graph **G**.

$$\text{cut}_{<M,N>} = \sum_{v \in M, u \in N} A_{uv} \tag{4.7}$$

A min-cut of the graph is a cut that has a smallest cut size. The well known algorithm for solving min-cut problem is based on the *max-flow min-cut* theorem by Ford and Fulkerson [33]. The flow based algorithm for identifying graph cuts has been used to solve the community detection problem in large World Wide Web graphs [31, 32].

A key hinderance in applying the cut based algorithm for community detection in a multi-relational heterogeneous network setting is that the *link density* in different networks (data sources) are different. As a result one may get trivial partitions or suboptimal partitions. For example, consider a multi-relational heterogeneous network $\mathcal{G}$, whose adjacency matrix

is constructed as follows

$$G_{ij} = \begin{cases} A_{ij} & i, j = 1, 2, ...n, \\ C_{ij} & i = 1, 2, ..n \text{ and } j = n+1, ..m \\ C_{ji} & j = n+1, ..n+m \text{ and } i = 1, ...n \\ B_{ij} & i = n+1, ..n+m \text{ and } j = n+1, ..n+m \ . \end{cases} \tag{4.8}$$

If the link density in bipartite graph $\mathcal{G}_{12}$ is very low, then the cut based algorithm would chop the multi-network into individual networks and subsequently split each of the individual networks independently, thus ignoring the valuable information given in the bipartite graph $\mathcal{G}_{12}$. On the other hand, if the link density within one of the two networks is much higher than the other (including the bipartite graph), then the cut based algorithm would treat it as a dense community and split the other network repeatedly. This is illustrated graphically in Figure 4.1. Such problems would not arise in the joint learning framework as the proposed objective function factorizes each network separately into their respective communities (and classes) and these latent factor variables in turn induce partition on the bipartite graph that link the two networks. Nevertheless, the link density does affect the working of the update formula in two ways. If the link density in one of the network is higher than the other, then it contributes more to the objective function than the other network. For example, in the objective function (4.1), if the matrix A is denser than the other two matrices, then more weights would be given in minimizing the first term making the algorithm more focussed in determining a better estimate of value X than for Y. This problem can be addressed by appropriately weighting the terms in the objective function or scaling the adjacency matrices.

4.3 Experimental Evaluation

This section presents the results of applying the proposed framework to the multi-task learning problem on both synthetic and real-world network data. We have designed a multi-network simulator to generate the synthetic data for our experiments. In addition, the Wikipedia and Digg data discussed in Chapter 3, are used to evaluate the performance of the proposed multi-task learning framework.

4.3.1 Baseline Algorithms and Evaluation Metrics

As a baseline, we use the normalized cut (`Ncut`) algorithm by Shi and Malik [91] for clustering and the label propagation algorithm with local and global consistency (`LGC`) by Zhou et al. [114] for classification. For a fair comparison, we applied each baseline algorithm on the entire multi-network $\mathcal{G}$ (instead of $\mathcal{G}_1$ and $\mathcal{G}_2$ separately). This is accomplished by constructing a single adjacency matrix G for the entire network using (4.8). In each experiment, we set the proportion of labeled nodes for the classification problem in one of the two network to be 0.2. The label factor β is set to 0.5 for synthetic data and set to 50 for Wikipedia, Digg data. We use the normalized mutual information (NMI) measure to evaluate clustering results and accuracy to evaluate classification results. Since we perform the classification task as semi supervised clustering, the accuracy measure is computed differently. Here , each estimated cluster is assigned to the ground truth class to which is most frequent in the cluster. If rows of the confusion matrix represents the ground truth class and columns represent the estimated clusters, then we sum the maximum values across each column and divide by total number of points. Note that if the confusion matrix is diagonal heavy (maximum of each row/column occurs in diagonal) then this measure is same as regular accuracy measure.

We denote our proposed multi-task, multi-network learning framework as `Joint` or `Joint + Prior` in the remainder of this section. Since the iterative update formula converges to a locally optimal solution, the `Joint` and `Joint + Prior` algorithms were run several times, each with different random initialization values for the pseudo-label matrices. We report the results produced by the run which minimizes the objective function (4.1) (instead of choosing the run that maximizes the accuracy of inferring the class and cluster membership of the nodes). For the synthetic data, we show the scatter plot of all the values from 15 different runs and highlight the values of the run with minimum error. We denote this run as `Joint(ME)`

4.3.2 Synthetic Data

Our multi-network simulator constructs two networks, $\mathcal{G}_1$ and $\mathcal{G}_2$. The parameters for constructing each network is summarized in Table 4.2 below. We assume there is a correspon-

TABLE 4.2: Parameters of multi-network generator

Parameter	Explanation
k_1	Number of clusters/classes in $\mathcal{G}_1$
P_{11}	Probability of within cluster link in $\mathcal{G}_1$
P_{12}	Probability of between cluster link in $\mathcal{G}_1$
k_2	Number of clusters/classes in $\mathcal{G}_2$
P_{21}	Probability of within cluster link in $\mathcal{G}_2$
P_{22}	Probability of between cluster link in $\mathcal{G}_2$
Q_1	Probability of link between nodes in two corresponding clusters in $\mathcal{G}_{12}$
Q_2	Probability of link between nodes in two non-corresponding clusters in $\mathcal{G}_{12}$

dence between the clusters/classes in $\mathcal{G}_1$ and the clusters/classes in $\mathcal{G}_2$. A bipartite graph $\mathcal{G}_{12}$ is also constructed by linking the nodes in $\mathcal{G}_1$ to those in $\mathcal{G}_2$. By default, each cluster

TABLE 4.3: Configurations of various synthetic data

Configuration	Network $\mathcal{G}_1$			Network $\mathcal{G}_2$		
	k_1	P_{11}	P_{12}	k_2	P_{21}	P_{22}
Noisy Network	4	0.30	**0.05 - 0.25**	4	0.20	0.10
Noisy Bipartite	4	0.20	0.12	4	0.20	0.15
Uneven	**3**	0.20	0.12	**5**	0.20	0.15

Configuration	Links in bipartite network $\mathcal{G}_{12}$	
	Q_1	Q_2
Noisy Network	0.12	0.03
Noisy Bipartite	0.30	**0.05 - 0.25**
Uneven	0.30	**0.05 - 0.25**

contains 100 nodes (so a network with 4 clusters will contain 400 nodes).

In this thesis we investigate three main configurations of the network parameter settings. The configurations are given in Table 4.3. The "Noisy Network" configuration varies the proportion of noisy links (i.e., P_{12}) in $\mathcal{G}_1$ while fixing all other parameters. The "Noisy Bipartite" configuration varies the proportion of noisy links (i.e., Q_2) in the bipartite graph. In both configurations, there is a one-to-one correspondence between the clusters/classes in the two networks. For the "Uneven" configuration, the networks contain different number of clusters/classes. We have 5 clusters in $\mathcal{G}_1$ and 3 clusters in $\mathcal{G}_2$. Thus, a many-to-one correspondence is established between clusters/classes in $\mathcal{G}_1$ to those in $\mathcal{G}_2$.

We evaluate the performance of our algorithm on the synthetic data set under four configurations that are listed in Table 4.3. These four configurations evaluate different aspects of learning multiple related networks.

4.3.2.1 Varying Noisy Links Within a Network

We use the "Noisy Network" configuration in which we vary the inter cluster noise (P_{12}) in network $\mathcal{G}_1$ from 0.05 to 0.25 (with a step size of 0.05). we the performed clustering on $\mathcal{G}_1$

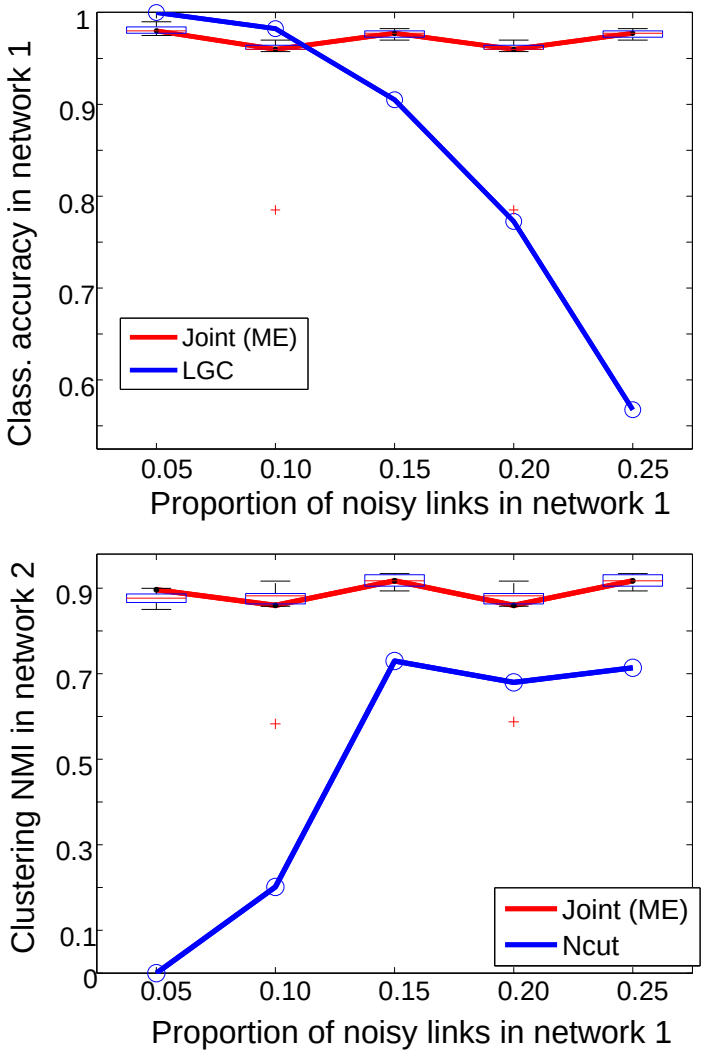

Figure 4.2 Effect of varying the between-cluster link probabilities (P_{12}) on $\mathcal{G}_1$. Top: Accuracy on $\mathcal{G}_1$. Bottom: NMI on $\mathcal{G}_2$

(classification on $\mathcal{G}_2$) and classification on $\mathcal{G}_1$ (clustering on $\mathcal{G}_2$) to study the effect of noise on both the learning task.

Classification on $\mathcal{G}_1$ Figure 4.2 compares the results for `Joint` learning against the `LGC` and `Ncut` on multi-graph. The top panel shows classification accuracy on $\mathcal{G}_1$ and the bottom panel shows the NMI of clusters obtained from $\mathcal{G}_2$.

When the noise level is low, the `LGC` algorithm performs slightly better than `Joint`. However, at higher noise levels, the performance of `LGC` drops significantly and the joint

learning performs much better. This result suggests that the explicit community information (from variable X) in $\mathcal{G}_2$ helps to discern between the classes in $\mathcal{G}_1$, whereas `LCG` is unable to use such information. Notice the inter quartile range for the box plot of accuracy values is very small, suggesting the robustness of the classification problem against the random initializations.

Surprisingly, as the noise level in $\mathcal{G}_1$ increases, the NMI of the `Ncut` algorithm also increases in $\mathcal{G}_2$. This is because of difference in the link densities between the two networks. (See Section 4.2.2). When $P_{11} = 0.3$ and $P_{12} = 0.05$, $\mathcal{G}_1$ has four distinct clusters, while in comparison, the network $\mathcal{G}_2$ with high link density (with P_{21}=0.20, P_{22}=0.10) by itself appears as a single community in the multi-network. The `Ncut` algorithm tries to identify four communities in the multi network. Due the difference in the link density between the two networks, the `Ncut` algorithm assign the entire $\mathcal{G}_2$ as one community and partitioned the network $\mathcal{G}_1$ into three communities. This results in lower NMI value on $\mathcal{G}_2$. The increase in the noise level in $\mathcal{G}_1$ increases the link density in the network resulting in identifying better cuts by the `Ncut` algorithm on the multi-network. When the noise parameter is $P_{12} = 0.15$, the link densities in both the network becomes comparable and the proposed `Joint` algorithm still performs better than `Ncut`.

Clustering on $\mathcal{G}_1$ For the same "Noisy Network" configuration , we performed community detection $\mathcal{G}_1$ and classification accuracy in $\mathcal{G}_2$. The results are shown in Figure 4.3. In both the cases, the `Joint` performs better than the respective baseline algorithms. As explained earlier, when the noise level is low the `Ncut` on multigraph identifies three communities on network $\mathcal{G}_1$ and the whole of $\mathcal{G}_2$ as fourth community. This resulted in NMI of 0.75. As the noise exceeded certain threshold ($P_{12} > 0.15$), the `Ncut` on multi-graph identified cuts in network $\mathcal{G}_2$ instead of $\mathcal{G}_1$. This resulted in decrease of NMI value in $\mathcal{G}_1$.

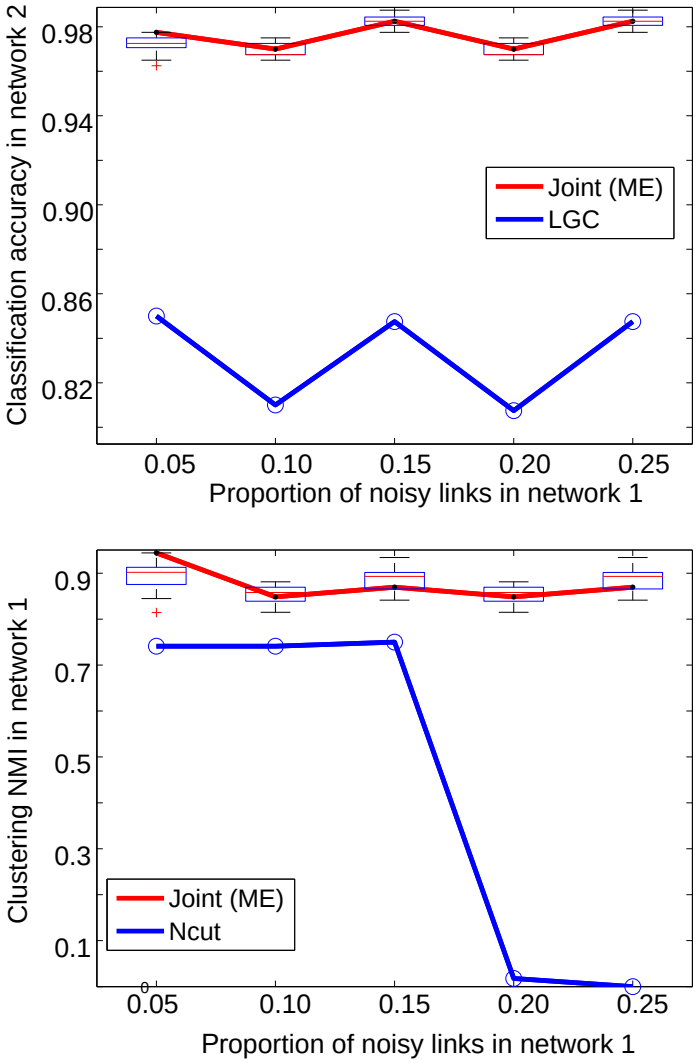

Figure 4.3 Effect of varying the between-cluster link probabilities (P_{12}) on $\mathcal{G}_1$. Top: accuracy on $\mathcal{G}_2$. Bottom: NMI on $\mathcal{G}_1$

Since the parameters for $\mathcal{G}_2$ are not varied, the classification results on $\mathcal{G}_2$ from `LCG` remains almost constant at 0.85 while the classification accuracy from `Joint` also remains pretty much constant at 0.98.

4.3.2.2 Varying Noisy Links Between Networks

We use the "Noisy Bipartite" configuration and vary q_2 from 0.05 to 0.25 in step size of 0.05. Varying the noise between the network will have no bearing on the independent clustering

and classification tasks on either network. Here again, we performed both clustering and classification on the multi-relational network $\mathcal{G}$ given by (4.8). We then recorded the classification accuracy on subgraph $\mathcal{G}_1$ and clustering NMI on subgraph $\mathcal{G}_2$. The results are shown in Figure 4.4. At lower noise levels, the `Joint` classification outperforms both `LGC` and `Ncut`. For higher noise levels in $\mathcal{G}_{12}$, the learned class information from $\mathcal{G}_1$ are not successfully transferred to $\mathcal{G}_2$ for community detection and vice versa. In terms of the objective function, the adjacency matrix C influences both the cluster membership(X) and class membership(Y) factors. As C becomes noisier, it propagates noise into factors X and Y, thus bringing down both accuracy and NMI.

4.3.2.3 Effect of Unequal Number of Clusters and Classes

We use the "Uneven" configuration by varying q_2 from 0.05 to 0.25 in step size of 0.05. We apply `LGC` to the entire multi-network $\mathcal{G}$ (with number of classes equal to 3) and `Ncut` to $\mathcal{G}$ (with number of clusters equal to 5). We compare their performance against the classification accuracy of `Joint` on $\mathcal{G}_1$ and NMI of `Joint` on $\mathcal{G}_2$. The results are shown in Figure 4.5. Here again, the proposed `Joint` learning algorithm outperforms `LGC` and `Ncut` at lower noise levels. The performance of the `Joint` is comparable to `LGC` and `Ncut` at higher noise levels.

4.3.3 Wikipedia Data

In this experiment, we perform classification task on the article network and the community detection task on the editor network. As mentioned earlier, there are four large communities in the editor network and 12 sub-topics in the article network. Here we perform four sets of experiments:

1. Apply `LGC` on article network only (with $k_1 = 12$) and `Ncut` on editor network only

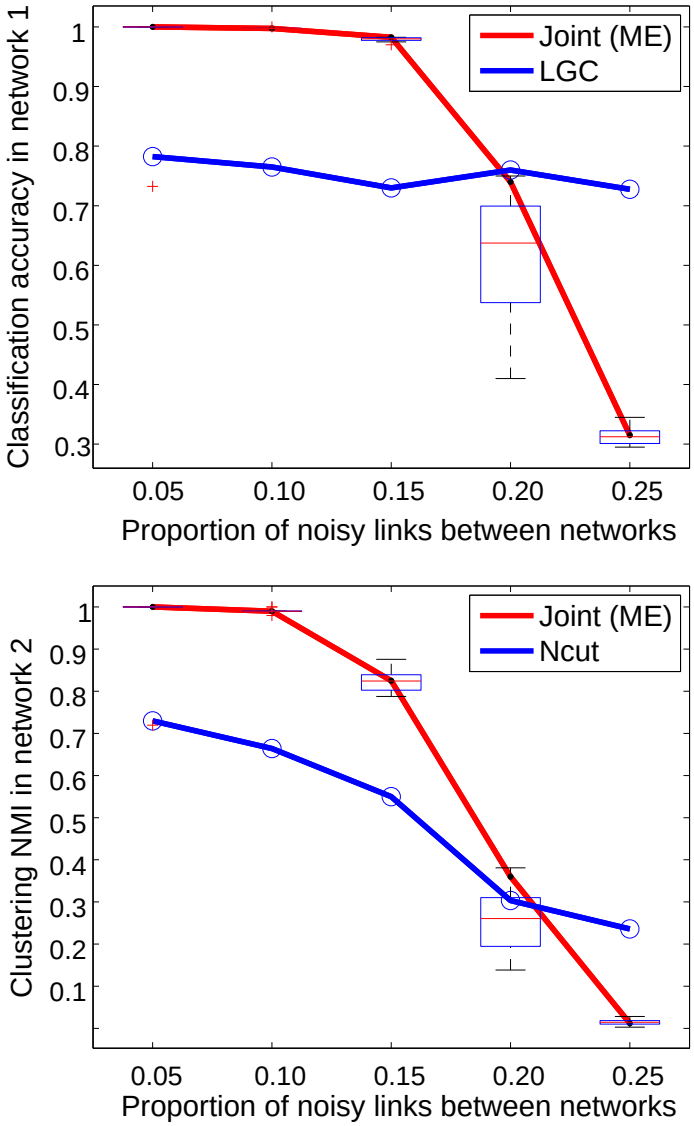

Figure 4.4 Effect of varying the between-cluster link probabilities (q_2) of bipartite network $\mathcal{G}_{12}$ with same number of clusters/classes in both $\mathcal{G}_1$ and $\mathcal{G}_2$. Top: accuracy on $\mathcal{G}_1$. Bottom: NMI on $\mathcal{G}_2$

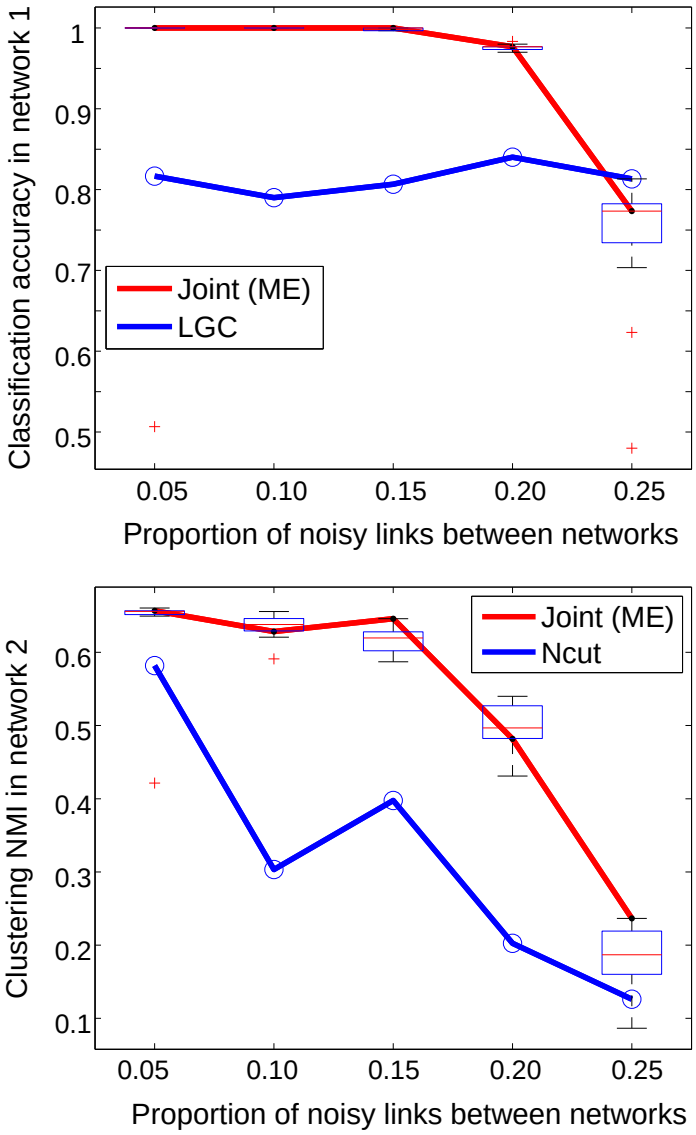

Figure 4.5 Effect of varying between-cluster Link probabilities (q_2) of bipartite network $\mathcal{G}_{12}$ with uneven number of clusters/classes. Top: Accuracy on $\mathcal{G}_1$. Bottom: NMI on $\mathcal{G}_2$

TABLE 4.4: Clustering results of Wikipedia editors. Here (ME) refers to the run with minimum error.

User network	4 clusters (NMI)
Ncut on editor network only	0.07
Ncut to entire multi-network	0.02
Joint(ME) without prior	0.32
Joint without prior	0.30 ± 0.0389
Joint(ME) with Prior	0.39
Joint with Prior	0.36 ± 0.0215

TABLE 4.5: Classification results of Wikipedia articles. Here (ME) refers to the run with minimum error.

Article network	12 classes (accuracy)
LGC on article network only	0.87
LGC on entire multi-network	0.85
Joint without prior (ME)	0.84
Joint without prior	0.80 ± 0.054
Joint with Prior (ME)	0.88
Joint with Prior	0.85 ± 0.021

(with $k_2 = 4$).

2. Apply `LGC` (with 12 classes) and `Ncut` (with 4 clusters) to the entire multi-network $\mathcal{G}$.

3. Apply `Joint` to the article network (with $k_1 = 12$) and editor network (with $k_2 = 4$) without using prior information.

4. Apply `Joint` to the article network (with $k_1 = 12$) and editor network (with $k_2 = 4$) with using prior information.

The results are shown in Tables 4.4 and 4.5. As shown in Table 4.4, the independent clustering of user network gives very bad results compared to the `Joint` approach. The cluster NMI increases from 0.07 to 0.32. Using the prior information further boosts NMI to 0.39. However, this additional gain comes at the expense of slightly reduced classification accuracy on the article network. Applying `LGC` on article network alone gives an accuracy

of 0.87 which reduces to 0.85, when applied to the entire multi-network. This is because the class information provided by the article network is more useful than the "coarse-level" cluster information provided by the editor network. Furthermore, a user typically contributes to articles across different categories which makes it difficult to decide his/her actual class label. We currently assigned the user to the category to which he/she has made the most contributions. In fact, it is because of this problem, it is difficult to acquire the label information in user network, and thus, clustering becomes a necessary task.

The `Joint` approach gives an accuracy of 0.84 on the article network. The loss in accuracy in the article classification can also be explained by examining the resulting confusion matrix, as shown in Table 4.6. The `Joint` approach identifies four communities in the editor network. Each of these four communities have major correspondence with three article sub-categories. Therefore, the `Joint` approach settles for a local minimum solution in such a way that the resulting confusion matrix is block diagonal instead of pure diagonal. That is to say, the identification of four communities in the editor network would propagate coarse-level cluster information from the editor network to article network. This makes it harder to discern the sub-categories in the article network and hence the drop in accuracy However, using the right prior information improves the accuracy to 0.88.

4.3.3.1 Number of Iterations

In this section ,we discuss the effect of number of iterations on the proposed joint factorization algorithm. Figure 4.6 shows the variation of the Wikipedia article network accuracy and the editor network NMI with every 10 iterations. Both the values tend to stabilize in long run. It should be noted that they do not necessarily exhibit a monotone property with respect to the number of iterations. It is not easy to determine the exact number of iterations required

TABLE 4.6: Confusion Matrix - Article Network using `Joint` algorithm

Political Science			Natural Science			Computer Science			Biology		
753	21	29	9	27	3	215	1	1	6	1	2
14	1064	27	32	38	11	96	1	8	29	2	0
1	6	622	7	0	3	25	9	0	4	7	4
0	4	2	680	1	17	1	0	3	1	0	0
11	2	0	41	514	4	16	5	0	13	0	1
1	7	0	16	0	747	1	0	1	7	0	0
0	0	0	2	0	2	22	102	5	1	1	0
0	1	0	3	0	2	27	420	1	0	0	0
3	80	0	8	0	6	7	3	769	0	0	13
1	4	14	1	0	4	4	0	0	345	32	24
0	0	2	2	0	2	1	0	0	17	870	37
1	7	11	1	0	3	5	1	0	139	49	589

for convergence. We run the algorithm until the error between two successive iterations is less than a specified threshold or maximum of 1500 steps.

4.3.4 Digg Data

Here, we linked the Digg users with Wikipedia editors based on the similarity between the words in the bookmarked URLs and words in the edited Wikipedia articles. There are three clusters in each network. The adjacency plot for these two networks is shown in Figure 4.7.

We first performed Ncut on the Digg data alone and Ncut on the overall network (Digg + Wikipedia + links between them). The confusion matrix is given in Table 4.7. As can be seen in the adjacency matrix plot, the first two clusters are noisy and heavily interlinked. So we obtain only two predominant clusters.

We apply the `LGC` algorithm to propagate labels in the Wikipedia data set. The results are summarized in Table 4.8. The noisy Digg data has degraded the performance of `LGC` on the Wikipedia network. Propagating labels only on Wikipedia data set gives an accuracy of 0.71, which reduces to 0.66 when applied to the multi-network.

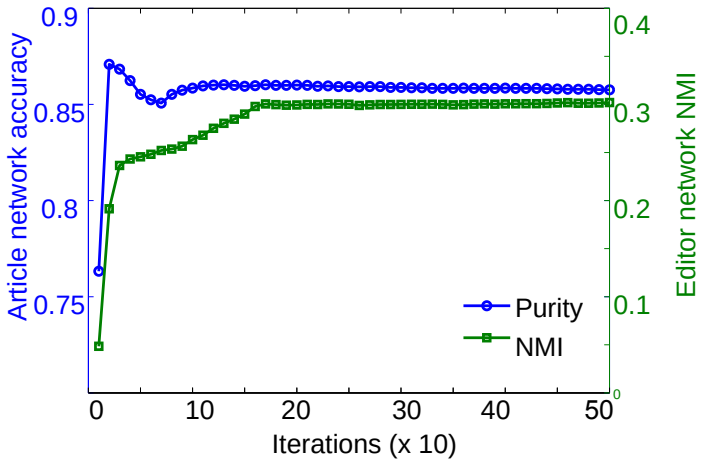

FIGURE 4.6: Plot showing the variation of classification accuracy and clustering NMI for every 10iterations (best viewed in color).

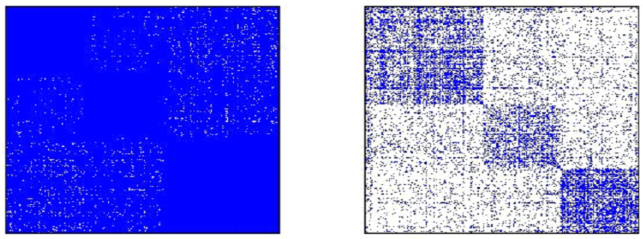

FIGURE 4.7: Adjacency matrix plot for Digg users (right) and Wikipedia editors (left) networks(best viewed in color).

The presence of noise on the Digg user network combined with noisy links between the Wikipedia and Digg networks resulted in poor performance of the joint learning algorithm. The number of clusters obtained is less than the number we expect. However, by incorporating the prior matrix $P = I_3$, this ensures that we obtain three clusters on each network. The results are shown in table below. Clearly, the `Joint + Prior` results are significantly better than both Ncut and LGC.

TABLE 4.7: Confusion matrix for Digg clustering. The top panel shows the result of `Ncut` on Digg user network. The bottom panel shows the clustering result on the multi graph consists of both Digg users and Wikipedia editors. Only two dominant clusters were found on both.

Ncut on Digg				Ncut on Digg + Wikipedia		
1171	3	474		167	1481	0
1149	0	486		997	638	0
392	0	1995		1952	435	0
NMI - 0.143				NMI - 0.185		

TABLE 4.8: Confusion matrix for Wikipedia classification. The left panel gives the confusion matrix on applying `LGC` on Wikipedia editor network and the bottom panel gives the confusion matrix on applying `LGC` on multi graph consisting of both Digg user network and Wikipedia editor network.

LGC on Wikipedia				LGC on Digg + Wikipedia		
1463	203	150		1688	78	50
350	690	159		710	416	73
232	100	859		450	56	685
Accuracy - 0.71				Accuracy - 0.66		

TABLE 4.9: Confusion matrix for `Joint` with prior information for the run with minimum error.

Digg - Cluster				Wiki - Classify		
1246	280	122		1710	84	22
136	1278	221		298	770	131
227	103	2057		55	118	1018
NMI = 0.44				Accuracy - 0.83		

4.4 Summary

In this chapter, we have given a framework to perform mutual learning on multiple related networks. Through various set of experiments, we infer that on a collection of noisy networks, multi task learning performs better than independent task learning on individual networks. We have also introduced the idea of using a prior to guide the clustering process. We have demonstrated a practical use of our algorithm by identifying similar communities on different network domains namely Digg and Wikipedia. Recently, researchers have used Wikipedia as knowledge source or auxiliary information source to perform several data mining operations like document clustering, web page classification, tagging and semantic relationship mining etc. [6, 38, 44, 101, 102]. In this thesis, we show yet another novel use of Wikipedia where in we use the link information between the Wikipedia articles to perform clustering of users in other networks. In particular, we use the link information in Wikipedia to cluster users in Digg network. Incorporating nodal attributes and improving scalability of the algorithm to networks having millions of nodes are two potential directions for future research.

Chapter 5

Joint Community Detection and Link Prediction

The ability to predict the formation of links in a network is an important task in network analysis. A reliable link prediction model is useful for uncovering missing links in a static network or for projecting the formation of new links in a dynamic network. The level of link prediction accuracy sought often varies depending on the context of its application. For example, despite its poor accuracy, the *FOF* (Friend of Friend) algorithm has been extensively used in predicting future links between users in large social networks. However, in critical applications such as bio-surveillance and terrorist network monitoring, predictions are cost sensitive in that there is a severe penalty factor associated with different incorrect predictions made by algorithm.

The low accuracies of link prediction algorithms can be attributed to the inherent *skewness* of network data. In this regards, there are two types of skewness to be considered. The obvious one is class skewness, which refers to the lopsided ratio of non-linked (the *negative* class) to linked (the *positive* class) node pairs in a network. Typically the ratio is of the order of $O(1/N)$. Such high skewness would result in a biased decision boundary and requires inclusion of skew correction approaches into the link prediction framework. Another type of skewness, which has received little attention in the link prediction literature, is in

the degree distribution of the nodes. Since many networks exhibit a scale-free behavior, this results in the (few) high degree nodes exerting the most influence on the prediction for the positive class. As a result, most link prediction models tend to fail in their prediction for the majority of the low-degree nodes. To avoid this, we need to develop a loss function for link prediction that considers both type of skewness in the data.

In addition, existing link prediction methods are either global or local in nature. The former (e.g., *common neighbors*[54]) simply utilizes information from the immediate neighborhood of the nodes to make its prediction. Though such an approach tends to perform poorly especially on large networks, it is computationally efficient. The latter (e.g., based on supervised learning [43, 85]) often achieves better performance but at the expense of higher computation time. Moving away from these two extremes is the method of finding links at the community (cluster) level. The intuition here is that links are more likely to be formed between nodes in the same community rather than those in different communities. However, identifying the right set of communities is itself a challenging problem. One of the most well-known community finding algorithms is based on the network modularity measure [72]. However, to the best of our knowledge, none of the existing link prediction algorithms are designed to optimize the measure.

The main contributions of this chapter are as follows:

- We propose a variable-cost loss function for supervised link prediction that considers both the imbalanced class distribution (of linked and non-linked node pairs) as well as skewness in the degree distribution. The variable-cost loss function addresses the bias in degree distribution by penalizing the misclassification of low-degree linked node pairs more than misclassification of high-degree linked node pairs.

- We show the intimate relationship between the proposed loss function and the modularity measure used to identify communities in a network. As a consequence, a link prediction algorithm that optimizes the proposed loss function is inherently biased towards finding links within communities without explicitly identifying the communities.

- We design a boosting algorithm for link prediction called `LinkBoost` that optimizes the cost-sensitive loss function. We also provide weak learners that utilize the nodal attributes to estimate the link potentials between the node pairs.

- We present an approach for scaling up the `LinkBoost` framework by first decomposing the network into smaller, potentially overlapping partitions and then combining the predictions made by the weak learners constructed from the different partitions.

To the best of our knowledge, the degree dependent cost sensitive link prediction algorithm is the first of its kind. Through the design of our loss function, the chapter also highlights the connection between community-based link prediction, modularity measure, and the boosting algorithm. Finally, experimental results show that our proposed LinkBoost algorithm *consistently* performs as good as or better than many existing methods when evaluated on 4 real-world network datasets.

5.1 Approaches for Link Prediction

Link prediction algorithms can be categorized in many ways. First, the algorithms can be supervised or unsupervised. Second, they can be based on the observed link structure only or may incorporate nodal attributes. Third, the algorithms may utilize the link structure information from immediate neighborhoods (local methods), entire network (global methods),

or at community levels.

The simplest unsupervised link prediction algorithm is based on computing the similarity scores between a pair of nodes using the nodal attributes or their local network topology. Examples of such local methods include common neighbors, Salton index [82], preferential attachment [7, 105], and Adamic-Adar index [2]. The performance of the different local measures were compared in [54, 116]. The results suggest that simple common neighbors approach performs better than other local measures. A theoretical justification for the better performance of *common neighbors* approach was presented in [78].

Unsupervised global methods for link prediction typically consider the weighted paths between node pairs. Examples include the Katz measure [49], random walk with restart [99], average commute time, and matrix forest index [15]. Measures based on paths, in general offer higher prediction accuracy compared to the local similarity measures. However, they require the entire network link structure and their computations are generally time consuming.

Link prediction using supervised learning has been investigated by many authors [43, 46, 85, 97]. Al Hasan et al. [43] derived several nodal and topological features for link prediction and applied a variety of classifiers such as support vector machines and decision tree to predict links in bibliographic databases. Kashima and Abe [46] proposed a parameterized probability model for the link structure and developed an expectation maximization algorithm to estimate the model parameters. Scripps et al. [85] employed a regularized matrix factorization approach for link prediction. Taskar et al. [97] used a relational Markov network to jointly model the nodal attributes and links. However, one of the main challenges in link prediction is the extremely large class skew, which leads to poor detection rate. Rattigan and Jensen [80] suggested an alternative problem known as anomalous link

discovery to identify the most interesting links in the network. Recently, there have been attempts to develop a semi-supervised approach for link prediction [48] as well as combining link prediction with other tasks such as collective classification [11].

More recently, there have been attempts to develop link prediction algorithms using generative models that account for the clustering (community) structure in the network. Guimera et al. [39], used the likelihood based methods for estimating the reliability of a link between any node pair, given the observed link structure. The reliability score is then used to predict both missing and spurious links. Clauset et al. [18] have proposed maximum likelihood based methods that represent the clusters in the network as a hierarchy, which in turn are represented as a dendrogram. Each dendrogram has an associated likelihood value indicating the strength of community structure represented by the dendrogram. The missing links are predicted by first sampling large a number of dendrograms proportional to their likelihood and for each unconnected node pairs i and j, the expected connecting probability is computed by averaging the corresponding probability over all sampled dendrograms. Finally, the node pairs are sorted according the connecting probability and highest ranked ones are declared as potential links.

Both the reliability and hierarchical cluster model try to estimate the link potentials between the node pairs at cluster level. They in fact, average over all possible partitions of communities present in given network which makes it is very costly to implement even on small sized networks. In this chapter, we suggest an alternative to these two algorithms which strive to identify more links with in a community. We do this by defining loss function whose associated risk when minimized, leans towards giving higher rating for the with in community node pairs. We also show the relationship between the proposed cost sensitive loss function and the well known modularity measure used for clustering networks.

5.2 Loss Function and Risk

We consider the link prediction task as a binary classification problem, in which a node pair is assigned to the positive class if there is a link between them, or to the negative class otherwise. Let $\mathcal{V} = \{1, 2, \cdots, n\}$ denote the set of nodes in the network and $\mathcal{E} = \mathcal{V} \times \mathcal{V}$ denote the set of all node-pairs. We represent the adjacency matrix of the network as A, where $A_{ij} = \{+1, -1\}$ indicating the presence or absence of links. Each node $i \in \mathcal{V}$ is associated with a set of d-dimensional nodal attributes $\mathbf{x}_i = \{x_{i1}, x_{i2}, ..., x_{id}\}$. Our objective is to learn a target function $f : \mathcal{V} \times \mathcal{V} \to \Re$ that maps each node pair to its link potential. The function is optimal if it minimizes the expected risk $R = E_{\mathcal{E},A}[L(f(e), a)]$ for any given node-pair $e \in \mathcal{E}$, where $L[f(e), a]$ is the loss function. The loss function usually takes the form of

$$L[f(\mathbf{e}_{ij}), A_{ij}] = \begin{cases} 0, & \text{if } \operatorname{sgn}(f(\mathbf{e}_{ij})) = A_{ij} \\ C_1, & \text{if } \operatorname{sgn}(f(\mathbf{e}_{ij})) \neq A_{ij} = 1 \\ C_2, & \text{if } \operatorname{sgn}(f(\mathbf{e}_{ij})) \neq A_{ij} = -1 \end{cases} \tag{5.1}$$

where $sgn(\cdot)$ is the sign function, whose value is equal to $+1$ if its argument is non-negative and -1 otherwise. When $C_1 = C_2 = 1$, this corresponds to the 0-1 loss function. Many supervised link prediction algorithms are designed to yield a classifier that minimizes the following 0-1 empirical loss function, which is given by

$$\hat{R}_{0\text{-}1} = \frac{1}{n^2} \sum_{i,j=1}^{n} \mathbf{I}\Big[A_{ij}\operatorname{sgn}(f(\mathbf{e}_{ij})) \leq 0\Big] \tag{5.2}$$

where $\mathbf{I}(\cdot)$ is an indicator function, which is equal to 1 if its argument is true and zero otherwise.

A major hinderance in this binary classification task is the class imbalance problem. In the social network data, negative examples (non-linked node pairs) tend to outnumber the positive examples by a significantly large proportion. The literature for classification on imbalanced data suggests two approaches to tackle this problem, namely, sampling and cost-sensitive learning. In the first approach, a balanced training set is obtained by undersampling the negative examples or oversampling the positive examples. This approach has several drawbacks. Firstly, undersampling the negative examples reduces the amount of data available for training an accurate model. Furthermore, one has to do the undersampling repeatedly to remove the sampling bias. On the other hand, oversampling the positive examples in the social network data increases the training set size significantly, which in turn, makes the training time considerably longer.

The cost sensitive learning approach is based on the premise that different classes of examples (positives or negatives) incur different penalties for misclassification. The loss function defined in (5.1) is cost sensitive if $C_1 \neq C_2$, where C_1 is the cost for misclassifying linked node pairs as non-linked node pairs and C_2 is the cost of misclassifying the non-linked node pairs as linked node pairs.

The loss function defined in (5.1) and the associated risk functions are not differentiable, hence does not offer mathematical dexterity in designing classifiers. The risk associlated with exponential loss can be used as an alternative:

$$\hat{R}_{\text{exp}} = \frac{1}{n^2} \sum_{ij} \exp \Big[- A_{ij} f(\mathbf{e}_{ij}) \Big] \tag{5.3}$$

The exponential risk is a continuous and differentiable function and it bounds the risk for 0-1 loss from above. It can be shown that an equivalent expression bounding the cost sensitive loss function defined in (5.1) is

$$\begin{aligned}\hat{R}_{\text{cost-sens}} &= \frac{1}{n^2}\sum_{ij}\Big[\mathbf{I}(A_{ij}=1)\exp(-C_1 f(\mathbf{e}_{ij})) \\ &\qquad + \mathbf{I}(A_{ij}=-1)\exp(C_2 f(\mathbf{e}_{ij}))\Big] \end{aligned} \tag{5.4}$$

By simply changing the class labels for presence and absence of links from $\{+1,-1\}$ to $\{C_1,-C_2\}$ the cost sensitive risk in Equation (5.4) can be transformed to the empirical risk of (5.3). Generally the cost parameters C_1 and C_2 are chosen in such a way that they correct for the classification bias that arises due to skewness in the class distribution. If n_+ and n_- represent the number of positive and negative examples in the data, then C_1 and C_2 are often chosen such that $\frac{C_1}{C_2}=\frac{n_-}{n_+}$. For large sparse networks, the fraction $\frac{n_-}{n_+}=O(n)$, thus if we fix $C_1=1$, then the value of $C_2\sim n^{-1}$ which results in working with extreme penalties that are easily polluted by the limitations of the machine precision. To avoid this, we need to scale the cost of both positive and negative labels such that the desired penalty ratio is maintained. Another significance of the cost ratio $\frac{C_1}{C_2}$ is its role in determining the optimal cost sensitive decision surface. The optimal decision surface for the cost sensitive learning

$$f^* = \arg\min_f E_{\mathcal{E},A}[L(e,a)]$$

is given by the Bayes Decision Rule [66]

$$f^*(e) = \log\frac{\mathrm{P}_{A|\mathcal{E}}(a=1|\mathbf{e})C_1}{\mathrm{P}_{A|\mathcal{E}}(a=-1|\mathbf{e})C_2} \tag{5.5}$$

Hence for any cost structure (C_1, C_2), cost sensitive optimality differs from cost insensitive optimality only through the threshold $T = \log \frac{C_1}{C_2}$.

The preceding formulation assumes that C_1 and C_2 are constants. We argue that it may not desirable to treat the misclassification cost for all the linked (and non-linked) node pairs by the same yard stick. In the next section, we present a variable cost loss function, such that misclassification of low degree linked node pairs incurs more penalty than misclassification of high degree linked node pairs. We also show that such modification leads to a link prediction algorithm that leans towards predicting more links within the same community than otherwise.

5.3 Variable Cost Loss Function for Link Prediction

This section describes our rationale for introducing a variable cost loss function for link prediction. It is generally observed that the degree distribution of real-world networks tends to follow a power law distribution, where there are few high degree nodes and a large number of low degree nodes. Consequently, a supervised learning algorithm for link prediction not only faces the bias from the large number of non-linked node pairs (negative class) but also from the small number of high degree nodes. Specifically, among the linked node pairs (positive class), the high degree nodes contribute more in determining the decision surface. Since we want to build models that can explain the observed links between any node pairs and not strongly influenced by the links formed for a few of the high degree nodes, we need to design a loss function that removes this bias within the positive class.

One way to do this would be to make the misclassification penalty dependent on the degree of the nodes. Let k_i be the degree of node i. Then the cost of misclassifying the

linked node pair $\mathbf{e}_{ij}$ is given by

$$C_1(\mathbf{e}_{ij}) = 1 - \beta k_i k_j,$$

where β is user defined parameter, typically chosen to keep the cost function non-negative. Notice that C_1 monotonically decreases with increasing degrees of k_i or k_j, thus penalizing more for misclassification of links between low degree nodes compared to misclassification of links between the high degree nodes.

Analogously, the same reasoning can be made about the non-linked node pairs. The low degree node pairs exert a higher influence on the negative class than the high degree node pairs. To remove this bias among the negative examples, we define the cost for misclassifying non-linked node pairs as

$$C_2(\mathbf{e}_{ij}) = \gamma k_i k_j,$$

which increases when the node degrees are higher. We now need to account for the overall bias between the positive and negative examples, this is done by choosing the value of β and γ such that $C_2 < C_1$. Putting it all together, we obtain the following loss function

$$L(f(\mathbf{e}_{ij}), A_{ij}) = \begin{cases} 1 - \beta k_i k_j, & \text{if } sgn(f(\mathbf{e}_{ij})) \neq A_{ij} = \ 1 \\ \gamma k_i k_j, & \text{if } sgn(f(\mathbf{e}_{ij})) \neq A_{ij} = -1 \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

The distinguishing aspect of the above loss function is that it assigns variable misclassification cost for different node pairs. When $\beta = \frac{1}{\sum_i k_i}$ the term $\beta k_i k_j$ represents the expected

number of links between the node pair i and j [72]. We will show in the next section that for this specific value of β, lowering the risk associated with the variable cost loss function is same as maximizing the modularity measure. This results in the learning algorithm being biased more towards learning links between the node pairs in same community than learning the links that lie between the communities.

5.4 Modularity

A well accepted conjecture in the network mining literature is that link densities are expected to be higher within a community than between communities. This suggests the possibility of an intimate connection between link prediction and community finding tasks. A popular method to identify communities in a network is using the well known modularity measure [72]. Here the possible existence of a community in a given network is revealed by comparing the actual link density in the subgraph induced by the community and the density one would *expect* to have if the nodes of the subgraph were linked irrespective of the community structure. The modularity measure can be mathematically quantified as follows.

$$\mathcal{Q} = \sum_{ij} \Big[\mathbf{I}(A_{ij} > 0) - P_{ij} \Big] \mathbf{I}(c_i = c_j), \tag{5.7}$$

where P_{ij} is the expected number of links between the nodes i and j under a null model (or reference network). The variables c_i and c_j represent the community membership of nodes i and j respectively. Modularity-based community finding algorithms are designed to assign the nodes to different communities such that the overall modularity measure, $\mathcal{Q}$, is maximized. The null model used often corresponds to that of a random graph with the same

degree distribution as the given network. This leads to [17]

$$\mathcal{Q} = \frac{1}{n^2} \sum_{ij} \left[\mathbf{I}(A_{ij} > 0) - \frac{k_i k_j}{2m} \right] \mathbf{I}(c_i = c_j) \tag{5.8}$$

where $m = \sum_i k_i/2$ is the number of links in the network.

The following theorem shows the equivalence between maximizing (5.8) and minimizing the risk associated with a special case of the loss function given in (5.6). Consider a community-based link prediction model that predicts the existence of a link between a node-pair based on whether the nodes are in the same community, i.e.,

$$sgn(f_{\text{comm}}(\mathbf{e}_{ij})) = \begin{cases} +1, & \text{if } \mathbf{I}(c_i = c_j); \\ -1, & \text{otherwise.} \end{cases} \tag{5.9}$$

Theorem 2 *For the variable cost loss function given in* (5.6)*, minimizing the risk associated with the community-based link prediction model* $f_{comm}(\boldsymbol{e}_{ij})$ *with* $\beta = \gamma = \frac{1}{\sum_i k_i}$ *is equivalent to maximizing the modularity function in* (5.8)

Proof 1 *The empirical risk associated with the variable cost loss function given in* (5.6) *for the community-based link prediction model is*

$$\begin{aligned} \hat{R}_{mod} &= \frac{1}{n^2} \Big[\sum_{ij:A_{ij}=1} \boldsymbol{I}(sgn(f_{comm}(\boldsymbol{e}_{ij})) = -1)(1 - \beta k_i k_j) \\ &\quad + \sum_{ij:A_{ij}=-1} \boldsymbol{I}(sgn(f_{comm}(\boldsymbol{e}_{ij})) = 1)(\gamma k_i k_j) \Big] \\ &= \frac{1}{n^2} \Big[\sum_{ij:A_{ij}=1} (1 - \delta_{ij})(1 - \frac{1}{2m} k_i k_j) \\ &\quad + \sum_{ij:A_{ij}=-1} (\delta_{ij} \frac{1}{2m} k_i k_j) \Big] \end{aligned} \tag{5.10}$$

where we have replaced $\mathbf{I}(sgn(f_{comm}(\boldsymbol{e}_{ij}))=1)=\delta_{ij}$ *and* $\beta=\gamma=1/2m$. *Now minimizing the empirical risk with respect to* δ *is equivalent to maximizing the following*

$$\begin{aligned}
& \min_{f} \hat{R}_{mod} \\
= & \max_{\delta} \frac{1}{n^2}\Bigg[\sum_{ij:A_{ij}=1}\delta_{ij}(1-\frac{k_ik_j}{2m}) - \sum_{ij:A_{ij}=-1}\delta_{ij}\frac{1}{2m}k_ik_j\Bigg] \\
= & \max_{\delta} \frac{1}{n^2}\sum_{ij}\delta_{ij}\Bigg[\boldsymbol{I}(A_{ij}>0)-\frac{k_ik_j}{2m}\Bigg] \qquad (5.11)
\end{aligned}$$

Since $\delta_{ij}=\mathbf{I}(sgn(f_{comm}(\boldsymbol{e}_{ij}))=1)=\mathbf{I}(c_i=c_j)$, *this completes the proof.*

The preceding theorem suggests that maximizing the modularity measure is equivalent to minimizing a special case of the loss function using the clustering solution, $f_{\text{comm}}(\mathbf{e}_{ij})$ as the link prediction model. The clustering solution uses only the network topology to explain the link potential between node pairs. In contrast, our proposed variable cost loss function provides a framework that allows us to estimate the link potential using other information including the nodal attributes. We design the $f_{\text{comm}}(\mathbf{e}_{ij})$ as function of nodal attributes x_i and x_j. Our experimental results have demonstrated the effectiveness of using an exponential loss compared to modularity function (5.24) for link prediction.

5.5 Boosting Approach for Link Prediction

This section presents our method for optimizing the variable-cost loss function given in Section 5.3. The risk associated with the loss function given in (5.6) is non-differentiable, so

we employ the following variable-cost empirical risk function:

$$\hat{R}_{mod} = \frac{1}{n^2}\sum_{ij}\Big[\mathbf{I}(A_{ij}=1)\exp[-(1-\beta k_i k_j)f(\mathbf{e}_{ij})] + \mathbf{I}(A_{ij}=-1)\exp[\gamma k_i k_j f(\mathbf{e}_{ij})]\Big] \tag{5.12}$$

If we set $\beta = \gamma$ then the above loss function reduces to

$$\hat{R}_{mod} = \frac{1}{n^2}\sum_{ij}\exp\Big[(\mathbf{I}(A_{ij}>0)-\beta k_i k_j)f(\mathbf{e}_{ij})\Big] \tag{5.13}$$

This form of loss function is well studied in the machine learning community using additive modeling or *boosting* techniques [19]. Specifically, an additive model takes the form of

$$f_\alpha(\mathbf{x}) = sgn\Big[\sum_t \alpha_t f_t(\mathbf{x})\Big].$$

For boosting, each f_i corresponds to a weak learner and the goal is to identify a sequence of constants $\alpha_1, ..\alpha_k$ such that a linear combination of the weak learners performs better than any of the individual learners.

5.5.1 Estimating α_t

Our aim is to design a boosting algorithm that minimizes the variable-cost empirical risk function $\hat{R}_{mod}$. To do this, we need to induce a sequence of weak learners that help in reducing the risk as optimally as possible. Let $F = \sum_{i=1}^{t-1}\alpha_i f^i$ be the previous solution of the boosting algorithm at step $(t-1)$ and f^t is the currently induced weak learner. We need to identify an appropriate α_t that would lead to an improvement in $\hat{R}_{mod}$. The optimization

problem at step t is given by

$$\min_{\alpha_t, f^t} \sum_{ij} \exp\left[-\left(\mathbf{I}(A_{ij} > 0) - \frac{k_i k_j}{2m}\right)\left(F_{ij} + \alpha_t f^t(\mathbf{e}_{ij})\right)\right] \tag{5.14}$$

To highlight the effect of current weak learner we need to isolate the effect of past weak learners from the equation. Let

$$\begin{aligned}
D_{ij} &= \exp\left[-\left(\mathbf{I}(A_{ij} > 0) - \frac{k_i k_j}{2m}\right) F_{ij}\right] \\
M_{ij} &= \left(\mathbf{I}(A_{ij} > 0) - \frac{k_i k_j}{2m}\right) f^t(\mathbf{e}_{ij}) \\
s_{ij} &= \text{sgn}(M_{ij}) \\
W^+ &= \sum_{ij \in M_{ij} > 0} D_{ij} \mid M_{ij} \mid \\
W^- &= \sum_{ij \in M_{ij} < 0} D_{ij} \mid M_{ij} \mid
\end{aligned} \tag{5.15}$$

It can be shown that the objective function given in (5.14) is bounded as follows:

$$\begin{aligned}
& \sum_{ij} \exp\left[-\left(\mathbf{I}(A_{ij} > 0) - \frac{k_i k_j}{2m}\right)\left(F_{ij} + \alpha_t f^t(\mathbf{e}_{ij})\right)\right] \\
= & \sum_{ij} D_{ij} \exp(-\alpha_t s_{ij} |M_{ij}|) \\
\leq & \sum_{ij} D_{ij} |M_{ij}| (\exp(-\alpha_t s_{ij}) - 1) \\
\leq & (W^+ \exp(-\alpha_t) + W^- \exp(\alpha_t) - W^+ - W^-)
\end{aligned} \tag{5.16}$$

where the inequality follows from applying Jensen's inequality and the assumption that

$|M_{ij}| \leq 1$. For a given f^t taking its partial derivative with respect to α_t gives

$$\alpha_t = \frac{1}{2} \log \frac{W^+}{W^-} \tag{5.17}$$

The formula for α_t is similar in spirit to regular AdaBoost in which $\alpha_t = \frac{1}{2} \log \frac{1-e}{e}$, where e is the error rate for the weak classifier. In our case, the W^+ and W_- represent the weighted sum of the correctly classified and incorrectly classified node pairs.

5.5.2 Weak Learners

This section describes the construction of the weak learners used in our boosting framework. Similar to traditional boosting, we could apply any simple classifier as long as it takes into consideration the weight matrix D associated with the node-pairs. The weak learner considered in this study is computed based on the nodal attributes and can be computed in closed form.

Let $\mathbf{X}$ represent the $n \times d$ nodal attribute matrix. Given the current weight matrix D between the node pairs, the goal of weak learner is to estimate the $n \times n$ link potential matrix $\mathbf{L(X)}$ where $L_{ij} = f^t(\mathbf{e}_{ij})$ indicates the strength of link between the nodes i and j. Large positive values of L_{ij} indicate greater potential for link between the nodes and large negative values indicate greater repulsion for link formation between the nodes. We model $\mathbf{L(X)}$ as simple weighted correlation of the nodal features. Let $\mathbf{L(X)} = \mathbf{XWX}^T$. Here the weight matrix $\mathbf{W}$ is a $d \times d$ matrix that needs to be estimated by solving the following objective function.

$$\mathcal{Q} = \max_W \sum_{ij} (D_{ij} B_{ij}) [XWX^T]_{ij}) - \frac{\lambda}{2} \parallel W \parallel_2^2 \tag{5.18}$$

where $B_{ij} = [\mathbf{I}(A_{ij} > 0) - \frac{k_i k_j}{2m}]$ is the coefficient term of the modularity measure or the cost associated with each node pair. Differentiating the objective function we get,

$$\frac{\partial \mathcal{L}}{W_{pq}} = -\sum_{ij} D_{ij} B_{ij} (X_{ip} X_{qj}^T) + \lambda W_{pq} = 0 \tag{5.19}$$

We get,

$$\begin{aligned} W_{pq} &= \frac{1}{\lambda} \sum_{ij} X_{ip} D_{ij} B_{ij} X_{qj}^T \\ &= \frac{1}{\lambda} \sum_{ij} X_{pi}^T D_{ij} B_{ij} X_{jq} \end{aligned} \tag{5.20}$$

Let $\bullet$ denote the element wise matrix multiplication, then W can be written as

$$\mathbf{W} = \frac{1}{\lambda} \mathbf{X}^T (B \bullet D) \mathbf{X} \tag{5.21}$$

Thus the link potential function $\mathbf{L(X)} = \mathbf{X W X}^T$ for the given weight matrix D is given by

$$\mathbf{L} = \frac{1}{\lambda} \mathbf{X X}^T (B \bullet D) \mathbf{X X}^T \tag{5.22}$$

A crude interpretation of the above solution is that it aligns the correlation between the nodal attributes with the modularity matrix. λ is chosen as a normalization constant such that the estimated link potentials are mapped between $[-1, 1]$. A distinct aspect of above definition of weak learners is that it does not require explicit conversion of nodal features to edge features. Traditional classifiers like support vector machine or logistic regression requires one to construct feature for each node pair from the nodal features, which itself is a time consuming process.

5.5.3 Scalability

Link prediction algorithms such as *Preferential Attachment* and *Common Neighbors*, though often have poor performance, are still considered attractive for many practical applications as they are easy to implement and scalable to large sized networks. Scalability is one of the important aspects of the proposed link prediction algorithm. Even though the number of links in large sparse networks is small, the supervised link prediction algorithm must examine all possible node pairs thereby increasing the size of data to be dealt with. In addition to the number of node pairs, the number of features associated with each node may add severe constraints on the performance of the model with respect to speed, memory requirement, and accuracy.

In this section, we describe an approach to scale up our proposed algorithm by decomposing the network into smaller, potentially overlapping partitions and using the boosting approach to systematically combine the weak learners constructed from each partition. This divide-and-conquer strategy is well suited both for the link prediction problem and the boosting framework since link formation is typically a local phenomenon, in the sense that there are several small communities in the network and the links are formed more inside that community. Thus it is beneficial to construct a local (weak) learner from a small segment of the network at a time and aggregate them in a principled way to form the global model via the boosting formulation.

There are many strategies to create subgraph partitions from a large network. Our requirements are that (1) the partitions must be distinctive enough from each other to induce a diverse (uncorrelated) set of weak learners and (2) the partitioning approach must be efficient to implement especially for large-scale networks. We tried several partitioning

strategies (e.g., applying random walk starting from randomly chosen seed nodes) but found that they often fail to satisfy one of the two requirements. This led us to consider the domain partitioning strategy, which is inexpensive to implement and often produces a diverse set of partitions.

The proposed scalable `LinkBoost` algorithm is summarized in Algorithm 2. The function `GetFeaturePartition`(η) returns a feature partition where each partition set contains $\eta\%$ of the features. For each partition, we create a subgraph containing only those nodes that have at least one non-zero value with respect to the selected set of features. We then build a local model on the subgraph by invoking the `GetBaseLearner` subroutine. The subroutine takes the following parameters as input: (1) A_v, the adjacency matrix associated with the subgraph induced by the feature partition P, (2) X_p, subset of the nodal attributes in the subgraph, (3) D_v, weights on node pairs in the subgraph, and (4) $\mathbf{k}_v$, global degree of the nodes in the subgraph. The weight returned by the `GetBaseLearner` subroutine is used to update the estimated link potential matrix. This process is repeated T times on all subgraphs obtained by different feature partitions.

In addition to its efficient implementation and diversity of its induced weak learners, another advantage of the domain partitioning strategy is that the final hypothesis has a nonlinear decision surface. It can be easily seen that the weak learner XWX^T described in Section 5.5.2 yields a linear decision surface separating the linked and non-linked node pairs. Since the boosting algorithm combines the weak learners also in a linear fashion, it will not be able to significantly alter the decision surface. However, by employing domain partitioning in the weak learner construction, we will work with a distinct subgraph at a time. The weight matrix W returned by `GetBaseLearner` function is applied only to the current subgraph V and not to the entire network. This results in inducing a non-linear

Algorithm 2 `LinkBoost`

Input: A: $n \times n$ adjacency matrix with $\{+1, -1\}$ entries
X: $n \times d$ nodal attribute matrix
η: threshold for feature partitioning
Output: **F**: $n \times n$ link potential matrix
Initialize:
F $= [0]_{n \times n}$; $D^{(0)} = [1]_{n \times n}$;
k: $n \times 1$ column vector of node degrees
for $t = 1$ to T **do**
$\mathcal{P} \leftarrow$ `GetFeaturePartition`(η)
for $X_p \in \mathcal{P}$ **do**
$V \leftarrow$ `GetSubGraphNodes`(X_p)
W $\leftarrow$ `GetBaseLearner`$(X_p, D_v, \mathbf{k}_v, A_v)$
Compute $f_v = X_p W X_p^T$
Compute α using (5.17)
$\mathbf{F}_v \leftarrow \mathbf{F}_v + \alpha f_v$
$D_v = D_v \exp(-\alpha A_v \bullet \mathbf{L}_V)$
end for
end for
`return` **F**

decision surface (clipped line) in the feature space (a line with respect to node pairs in current partition and value zero for node pairs outside the partition). Finally the boosting algorithm combines the collection of clipped lines to produce a final classifier with non-linear decision surface.

5.6 Experimental Evaluations

This section reports the results of experiments conducted on the proposed `LinkBoost` algorithm. Since link prediction is cost sensitive in nature, we compare the algorithm against other baseline methods using the receiver operating characteristic (ROC) curve. The curve is obtained by calculating the true positives and false positives by varying the threshold on the estimated link potentials between the node pairs. The link prediction model is built on the training set while the ROC curves are plotted for the node pairs in the test set.

5.6.1 Baseline Algorithms

We compared the performance of `LinkBoost` against the following link prediction algorithms discussed in the related work section.

Link-Based: We used three link based algorithm for link prediction. These are *Preferential Attachment*, *Katz* and *Modularity*. Preferential attachment estimates the link potential between a node pair as product of their degrees. The Katz measure is defined as

$$\texttt{score}(x, y) = \sum_{l=1}^{\infty} \beta^l \mid path_{ij}^{(l)} \mid \tag{5.23}$$

where $\mid path_{ij}^{(l)} \mid$ is set of all path of length l from node i to node j and $0 < \beta < 1$ is a user parameter. A special variant of Katz is the truncated Katz in which only finite number of terms in the summation are considered. The number of terms to consider is again a user given parameter. The Katz measure is sensitive to both these parameters. The modularity measure for link prediction is computed as follows. Let S_{ir} to be 1 if vertex i belongs to group r and zero otherwise. Then modularity maximization involves identifying a $n \times k$ matrix $\mathbf{S}$ with elements S_{ij} such that following equation is maximized.

$$\max_{\mathbf{S}} \operatorname{tr} \mathbf{S}^T \mathbf{B} \mathbf{S} \tag{5.24}$$

The problem (5.24) is NP hard and is relaxed by letting $\mathbf{S}$ to be any real matrix such that $\mathbf{S}^T\mathbf{S} = \mathbf{I}$. We then define $f_{comm}(e_{ij}) = \sum_r S_{ir} S_{jr}$.

Attribute Based: Here we use the well known *Fixed Cost Adaboost* where the cost parameters are set to $C_1 = 1$ and $C_2 = 0.01$. It is not possible to make the cost more than 1 as the derivation of α assumes that $|M_{ij}| \leq 1$. Furthermore, only the cost ratio $\frac{C_1}{C_2}$

matters and not their absolute magnitudes. Similarly, the modularity matrix **B** is multiplied by constant factor so that the magnitude of entries are less than 1. For `LinkBoost`, we set $\eta == 0.05$. The base learners used for both `LinkBoost` and *Fixed Cost Adaboost* are the same (see Section 5.5.2).

5.6.2 Data Sets for Inferring Missing Links

Here we consider the problem of inferring missing links from an incomplete network. We use two well-known citation networks [1] [89]—citeseer and cora data sets—for this experiment. In both the data sets, we first make the graph undirected and randomly suppress 30% of the links from the network and use them as the test set for predicting missing links.

Cora Data Set contains publications from the machine learning area, which include the following 7 subcategories: Case-based reasoning, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning and Theory. The data set we use contains 2708 nodes, 5429 directed links, and 1433 unique words. Each node corresponds to a paper and is characterized by a 0/1-valued vector indicating the absence/presence of the corresponding word from the title of the paper.

Citeseer Data Set consists of data from 3312 scientific publications. Each publication is labeled as one of 6 classes. The data set we have created contains 4732 links and 3703 unique words.

5.6.3 Data Sets for Predicting Future Links

Here, we are given the network link structure and the nodal attributes at a particular time period. Our task is to predict the link formed between the given nodes at a future time.

[1] http://www.cs.umd.edu/projects/linqs/projects/lbc/index.html

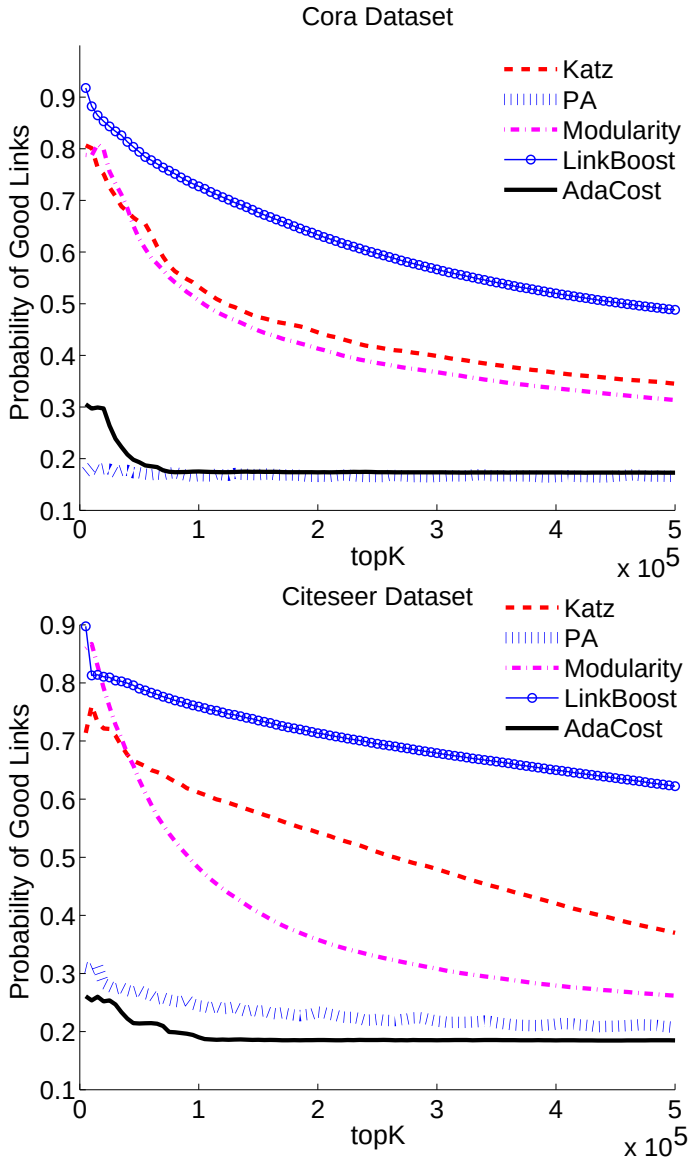

FIGURE 5.1: Proportion of within community links (good links) as function of topK values

TABLE 5.1: Link Prediction: The table shows the AUC of predicted missing links and future links in each of the four data sets.

		AUC (% improvement compared to LinkBoost)			
Data	Method	Cora	Citeseer	DBLP	Wiki
Link Only	Katz	0.72 (-16.70%)	0.63 (-41.30%)	0.61 (-17.50%)	0.86 (-2.20%)
	PA	0.63(-14.20%)	0.59(-33.70%)	0.71(-4.00%)	0.90(+2.20%)
	Modularity	0.67(-20.00%)	0.60(-32.50%)	0.63(-14.86%)	0.64(-27.27%)
Link+ Content	AdaCost	0.53 ± 0.09(-36.90%)	0.54 ±0.12 (-39.32%)	0.56 ± 0.2 (-24.32%)	<.50
	`LinkBoost`	**.84**± 0.025	**.89**±0.063	**.74**± 0.18	.88± 0.14

DBLP Data Set contains all the computer science articles [2] from the proceedings of 28 conferences related to machine learning, data mining and databases from 1997 to 2006. The train set consists of all publications from 1997-2000 and test set contains all publications from 2001-2004. There are 9252 nodes in the train set with 9136 nodal attributes. There are $21,107$ links in the train set and only 6679 links in the test set.

Wikipedia Data Set is a web page network which was crawled from Wikipedia web site by Kossinets[3]. The data set contains edit history of all the pages in Wikipedia from its inception until January 2008. We examined the user-user interaction network (user talk pages). The user interactions in the first 6 months of 2004 is taken as train set and the next 6 months is taken as test set. There are 8178 users and 24891 features for each user.

5.6.4 Links Within Community

First, we evaluated the performance of the LinkBoost algorithm in terms of its ability to predict links within community. For both cora and citeseer data sets we use the ground truth community label to verify the proportions of links formed within community for each of the link prediction algorithms. We sort the link potentials and declare the top-K largest link potentials as the possible missing or future links. Figure 5.1 shows the plot of the proportion of within community links or good links as function of top-K values. Clearly, the `LinkBoost` algorithm outperforms both modularity and Katz measures, thus validating the claim that our algorithm indeed strives to identify links within a community. The proportion of good links identified by modularity and Katz are quite high for smaller values of topK, but falls significantly for larger topK values.

[2] http://dblp.uni-trier.de/

[3] G. Kossinets. Processed Wikipedia Edit History. Stanford large network dataset collection.

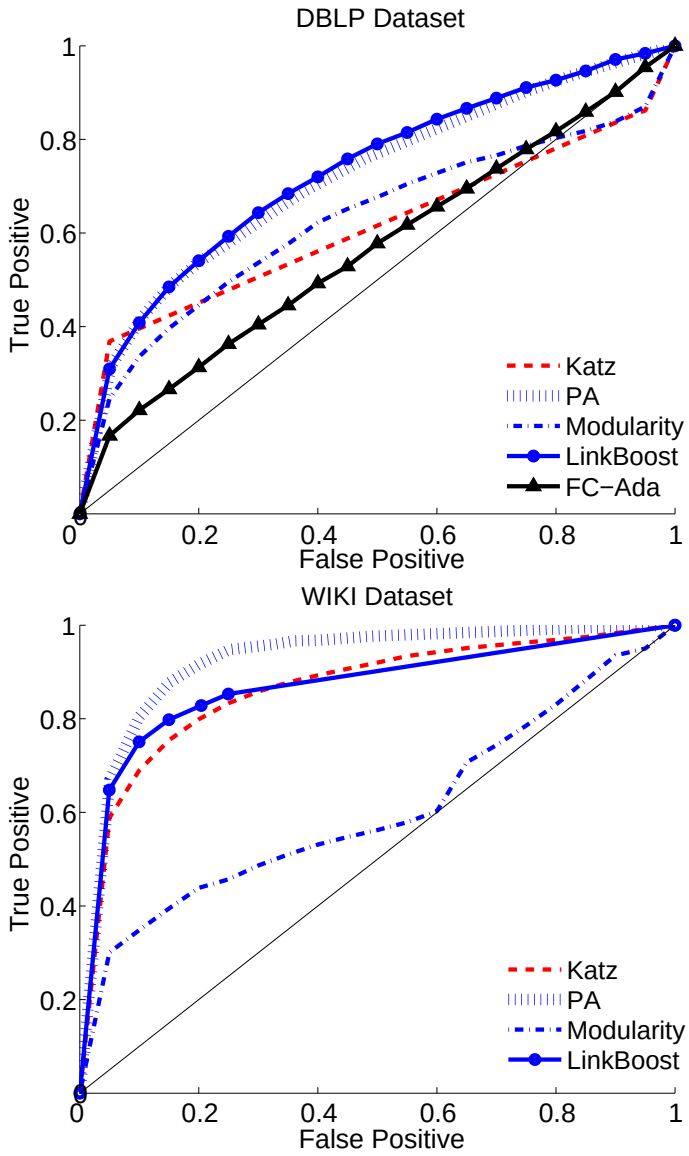

FIGURE 5.2: ROC curves comparing performances of different link prediction algorithms.

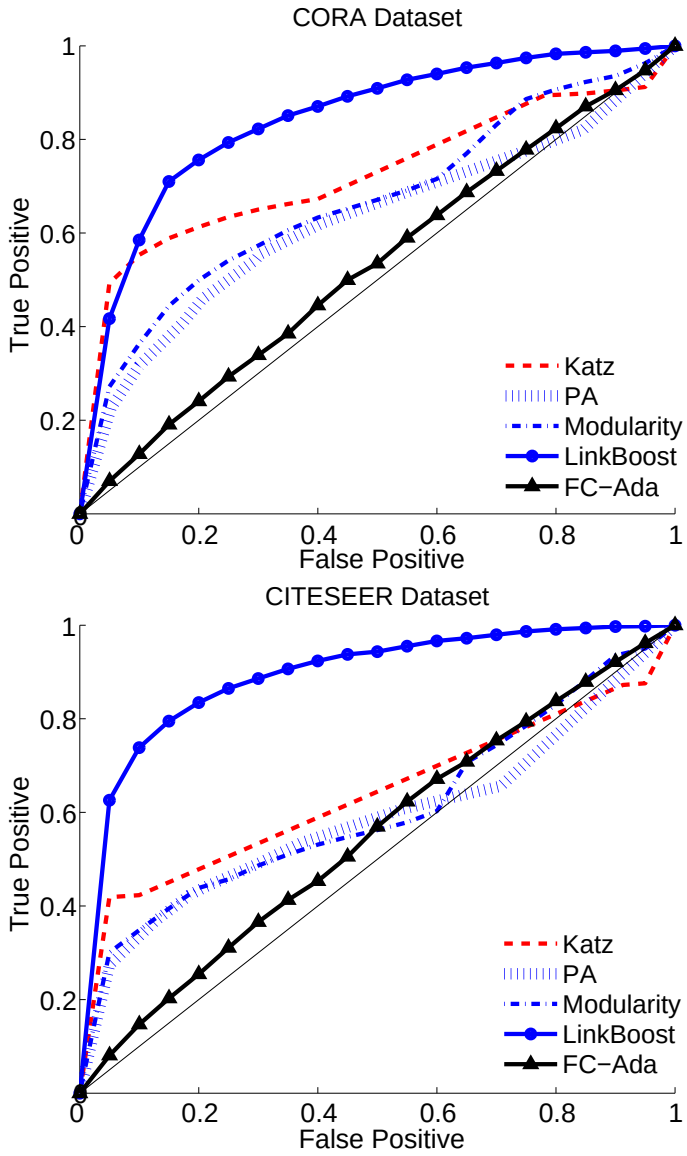

FIGURE 5.3: ROC curves comparing performances of different link prediction algorithms.

5.6.5 Missing and Future Links

Next we evaluate the performance of `LinkBoost` for the missing and future link prediction problems. The ROC curves are shown in Figure 5.2 and 5.3. Firstly, notice that the AUC of `LinkBoost` is consistently higher than *modularity* measure. As mentioned earlier, modularity utilizes only the network link structure whereas boosting makes use of both the link and the content information thus resulting in superior performance. The `LinkBoost` consistently outperforms the fixed cost Adaboost as well, highlighting the importance of the proposed variable cost structure.

`LinkBoost` outperforms the Katz measure on both cora and citeseer citation networks. The Katz measure performs better than the fixed cost Adaboost on the DBLP network and as good as `LinkBoost` on Wikipedia network. However it is sensitive to choice of parameter setting. In this chapter, we report the results based on the parameters that best fits the test set.

Finally, `LinkBoost` outperforms the preferential attachment measure on both the citation networks. However it is performance is comparable to preferential attachment on DBLP and Wiki networks. Specifically, `LinkBoost` is slightly better than preferential attachment on DBLP network and is slightly worse on Wikipedia network. This is because the preferential attachment algorithm is based on the premise that the *rich gets richer*. We suspect that the user network in Wikipedia exhibit the preferential attachment characteristics where few authoritative users communicate with large number of other users. The average AUC for `LinkBoost` is 0.88 and for preferential attachment is 0.90.

5.6.6 Low Degree Nodes

In this section, we demonstrate the ability of the proposed method to identify the links formed between low degree nodes in the citation networks. A node with degree less than 2 is considered to be a low degree node. We compute the models on the training set and estimate the ROC curves for the subgraph consisting of low degree nodes. The results are plotted in Figure 5.4. As expected, the preferential attachment measure under performs as it ranks the high degree nodes ahead of the low degree nodes. The proposed `LinkBoost` with effective degree sensitive loss function overcomes this problem.

5.7 Summary

In this chapter, we have given a new direction for the supervised link prediction problem in large sparse networks. We have proposed a new degree dependent cost function and has shown that minimization of the associated risk leads to modular link prediction where more links are predicted within community. Such a cost function addresses the skewness in class distribution and skewness in nodal degrees. The proposed algorithm is scalable and easy to implement. Experimental evaluations show the superior performance of the proposed method over existing supervised and unsupervised methods. The proposed method is specially effective in predicting the missing links for the low degree nodes. For future work, we plan to investigate methods for estimating optimal cost parameters and alternate ways for creating the weak learners used in LinkBoost formulation.

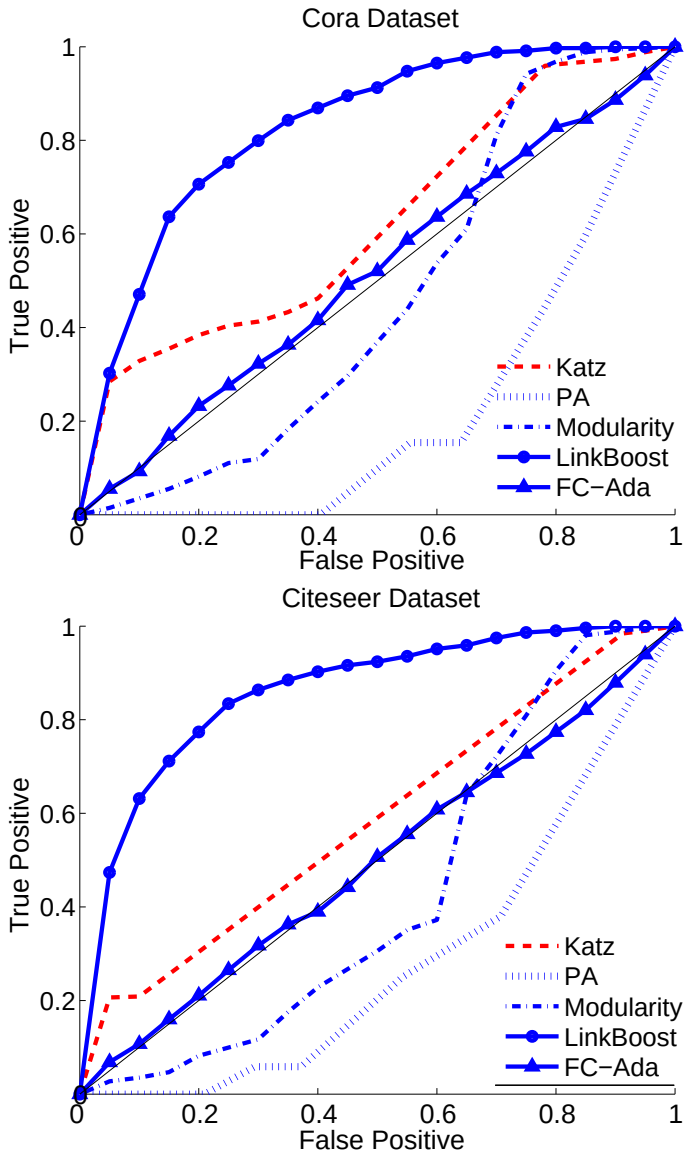

FIGURE 5.4: ROC curves comparing performances of different missing link prediction algorithmson the subgraph induced by the low degree nodes in the citation networks.

Chapter 6

Crowdsourcing for Network Mining

In the previous chapters, we have presented multi-source and multi-task network mining frameworks for both supervised and unsupervised learning tasks. A key requirement for developing supervised learning algorithms is the availability of trustworthy label information for the given mining task. The quality of the labeled data often affects the performance of the learning algorithm. In this chapter, we present a framework that engages the services of crowdsourcing technology in order to acquire (or augment) the label information for the network data. First, we briefly discuss the role of crowdsourcing in aiding the label acquisition process. Next, we highlight the challenges that are present in acquiring label information for network data. Finally, we present a novel approach which transforms the given network data into an image corpus for labeling by the crowd.

Crowdsourcing [23] is an emerging technology where a group of human workers, some of whom might be unskilled, are employed to solve a certain task that cannot be automatically and/or reliably solved by computers. For example, image annotation is a task that involves categorizing individual images in a corpus into certain pre-defined categories. It is not always possible to automate this task using computers and often times, humans can perform the task more accurately than computers. The key challenge for harnessing the power of the crowd lies in converting the problem at hand into a simpler task that can be handled by humans with great ease and speed. Such tasks are called Human Intelligence Tasks or HITs. For example, in the image annotation problem, the individual images constitute a HIT, which

are displayed to workers in order to elicit their label information. In the past, researchers have successfully employed the crowd to annotate data for search and retrieval problems [3, 67, 81, 93]. In addition, the power of the crowd has been harnessed to perform machine learning tasks such as clustering and classification [37, 45, 113].

One advantage of utilizing the services of crowdsourcing is that once the HITs are designed, they can be solved even by low skilled human workers (without any domain knowledge or expertise) for a menial payment. Therefore, the valuable time of domain experts can be spared from performing cumbersome data labeling task and focused on analyzing the data instead. The goal of this chapter is to design simple and easy to use HITs that can be presented to the crowd for acquiring label information in the network data. Unlike the image annotation problem, designing HITs for network data is more challenging as the raw network data does not easily lend itself to be presented to the crowd. Also, there are inherent privacy concerns when presenting social network data to third party workers. Therefore one has to design HITs that are simple and intuitive for average humans to act upon and at the same time do not disclose the original data in any way to the workers. To overcome this challenge, we present a data transformation approach whereby the network data is initially transformed into images so that it can visualized and subsequently classified by the crowd.

6.1 Transforming Network Data into Images

Data transformation is a preprocessing procedure for converting an input pattern into a suitable representation before we apply supervised and unsupervised learning algorithms. Traditional approaches are mostly designed for in-domain data transformation, which means they simply manipulate the input space in such a way that the transformed space is more

aligned with the requirements of the learning algorithm. Such approaches would typically project the data into a low dimensional manifold (in order to remove noise as well as to eliminate redundant and irrelevant features) or map the data into a higher dimensional space (to extend the feature representation and enable the use of linear classifiers to discern patterns that belong to different classes). The in-domain transformation approaches do not bring any new information that was not already present in the original data.

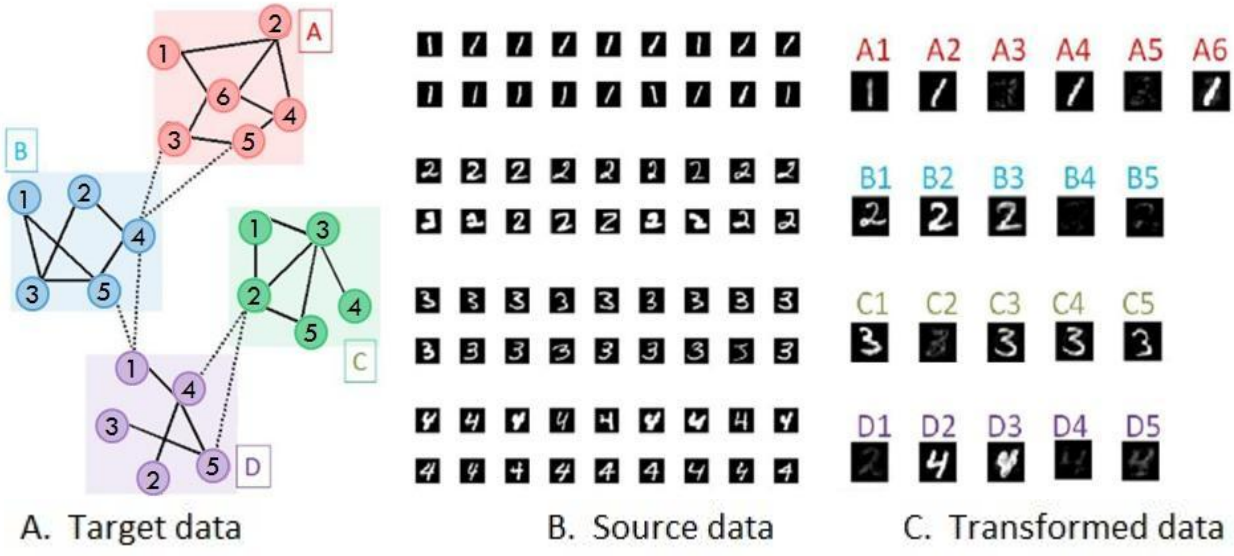

FIGURE 6.1: A toy example consisting of *target* network data (A) and *source* handwritten digit data (B). We map each distinct labeled target node to a unique source image and learn the transformation between target and source data. When this transformation is applied on all the target nodes, we get the transformed target data (C). The blurry images are interpreted in Section 6.1.

Unlike previous research, this work investigates an *out-of-domain* data transformation approach that enables the use of crowdsourcing technology for network mining problems. Specifically, the transformation involves two unrelated domains (*source* and *target*), each having a unique set of attributes, classes, and probability distributions. The target is the domain in which the desired classification task is to be performed but has limited labeled data, whereas the source serves as an auxiliary data source for which labels are already available or can be easily acquired even from non-experts. For this study, the source corresponds to a collection of labeled images whereas the target is a social network. Since the domains

are unrelated, a key challenge is to learn a proper transformation function that maps each labeled example in the target domain to its corresponding "surrogate" in the source domain.

To illustrate the proposed approach, consider the toy example shown in Figure 6.1. Here the source domain is a collection of hand written digit images corresponding to digits $1, 2, 3$ and 4 (see Figure 6.1B). The target domain, is a network data with four distinct communities denoted as A, B, C and D, as shown in Figure 6.1A. The circles and lines are the nodes and links, respectively. Solid circles correspond to labeled nodes, whereas unfilled circles represent the unlabeled ones. The solid and dotted lines represent within community and between community links, respectively. We map each distinct labeled node to one of the labeled images and learn their corresponding transformation matrix. The mapping is done in such a way that nodes from communities A, B, C and D are mapped to distinct images corresponding to digits $1, 2, 3$ and 4, respectively. The transformation is then applied to the unlabeled examples in the target domain to generate their corresponding images for subsequent labeling by the crowd. Figure 6.1C gives a pictorial representation of the transformed node representation in the image space. A good transformation may help reveal certain aspects of the underlying network. For example, the node $C2$ in community C has links to nodes outside of its community. After the transformation, even though the majority of the images associated with the nodes in community C are mapped to the digit 3, the image for node $C2$ is harder to discern because it lies at the border with other communities. Furthermore, the node $D1$, which has more links to nodes in community B than to those in its own community, is transformed into an image that resembles digit 2 more than digit 4. This example suggests that, although the feature space has completely changed, the transformed images should still retain useful information that helps provide insights into the latent structure of the network data.

A key question that remains to be answered is whether it is always possible to find a proper transformation that effectively maps nodes in a social network to images that can be easily discerned by humans. To help answer this question, we analyze the reconstruction error of the transformation and examine its relationship to the ranks of data matrices in the source and target domains. An alternating least square method is presented to learn a transformation matrix that minimizes the reconstruction error. Using network data from two real-world domains, we empirically showed the effectiveness of the framework in providing labels to solve a variety of network learning tasks including link prediction and node classification.

The remainder of this chapter is organized as follows. Section 6.2 presents the notations used in this chapter and defines the data transformation problem for crowdsourcing social networks. In Section 6.3, we introduce our proposed framework for augmenting training data in the target domain with newly acquired labeled data from the source domain. Section 6.4 describes the detailed methodology for learning the transformation. We also discuss the conditions under which an exact transformation can be found. Experimental results are given in Section 6.5. Finally, we conclude with a summary of the work and suggestions for future research.

6.2 Preliminaries

Let $\mathcal{S}$ and $\mathcal{T}$ denote the source and target domains. Throughout this chapter, we assume the source is an image corpus for which labels can be easily acquired from the crowd (human workers) whereas the target is a social network for which obtaining reliable labels is expensive. Both domains are assumed to have their own training examples. We further assume that

the classifier in the target domain can be improved if more labeled examples are augmented into the training set.

Let $\mathbf{X}^{(s)}$ be an $n_s \times d_s$ data matrix for the source domain and $\mathbf{Y}^{(s)}$ be its corresponding $n_s \times c_s$ class membership matrix, where n_s is the number of labeled examples, d_s is the number of attributes, and c_s is the number of classes. Each element y_{ij}^s in the matrix $\mathbf{Y}^{(s)}$ is equal to 1 if the labeled example $\mathbf{x}_i^{(s)}$ belongs to class j and zero otherwise. Similarly, let $\mathbf{X}^{(t)} = [\mathbf{X}^{(tl)}; \mathbf{X}^{(tu)}]$ denote an $(n_t + r) \times d_t$ data matrix for the target domain and $\mathbf{Y}^{(t)}$ be its corresponding $n_t \times c_t$ class membership matrix, where n_t is the number of labeled examples, r is the number of unlabeled examples, d_t is the number of attributes, and c_t is the number of classes in the target domain. For brevity, we assume $c_s = c_t = c$ and $n_s \gg n_t$.

Our goal is to learn a $d_t \times d_s$ transformation matrix $\mathbf{U}$ that effectively maps each labeled target example to a *surrogate* example in the source domain in such a way that preserves the distance and label information of the two data sets as much as possible. For example, in Figure 6.1, we seek a transformation matrix $\mathbf{U}$ that maps all the nodes from community A to images containing the handwritten digit 1, those from community B to images containing the digit 2, and so on. After learning the transformation, any unlabeled node can be mapped to its corresponding image by applying the matrix $\mathbf{U}$ to its node attributes. The transformed images can be labeled by human workers and the newly acquired labels can be combined with the original labeled data to train a better classifier.

6.3 Proposed Framework

This section presents an overview of our proposed framework for augmenting training data in the target domain with labeled examples acquired from the source domain. The framework

consists of the following tasks:

1. **Surrogate Mapping.** Given the labeled examples in the source and target domains, $(\mathbf{X}^{(s)}, \mathbf{Y}^{(s)})$ and $(\mathbf{X}^{(tl)}, \mathbf{Y}^{(tl)})$, we need to learn a transformation matrix $\mathbf{U}$ that maps each $\mathbf{x}_i^{(tl)} \in \mathbf{X}^{(tl)}$ to its surrogate $\mathbf{x}_j^{(s)} \in \mathbf{X}^{(s)}$. The mapping should be done in such a way that instances of a particular label in the target domain are mapped to instances of a fixed label in the source domain.

2. **Surrogate Labeling.** The transformation matrix $\mathbf{U}$ will be applied to unlabeled data in the target domain $\mathbf{X}^{(tu)}$ to generate new surrogates $\hat{\mathbf{X}}^{(su)}$ for labeling by human workers. Since each surrogate can be labeled by more than one workers, a consensus on the class label must be made for each target instance (e.g., by taking a majority vote on the class labels). Let $\mathbf{Y}^{(tu)}$ denote the consensus labels obtained for the unlabeled target instances.

3. **Model Building.** The newly labeled target examples $(\mathbf{X}^{tu}, \mathbf{Y}^{tu})$ are merged with the original training data. A classifier will be trained on the extended training data to generate a new predictive model for the target domain.

The key challenge is to develop an effective and efficient algorithm for learning the transformation matrix $\mathbf{U}$. We describe the details of the algorithm in the next section.

6.4 Surrogate Mapping

We cast the surrogate mapping task into a constrained optimization problem. First, we need to design an objective function that assigns each target example to a unique source example satisfying the label consistency requirement as well as minimizing the reconstruction error.

In order to do this, we define an $n_t \times n_s$ surrogate selection matrix $\mathbf{P}$, such that $\mathbf{P}_{ij} = 1$ indicates that the source example $\mathbf{x}_j^{(s)}$ is the surrogate for target example $\mathbf{x}_i^{(t)}$. The objective function for the surrogate mapping task is given below:

$$\begin{aligned} &\min_{\mathbf{U},\mathbf{P},\mathbf{Q}} \parallel \mathbf{P}\mathbf{X}^{(s)} - \mathbf{X}^{(tl)}\mathbf{U} \parallel_F^2 + \parallel \mathbf{P}\mathbf{Y}^{(s)}\mathbf{Q}^T - \mathbf{Y}^{(tl)} \parallel_F^2 \\ &\text{s.t.} \quad \forall i,j: \ \mathbf{P}_{ij} \in \{0,1\}, \quad \mathbf{P}\mathbf{1}_{n_s} = \mathbf{1}_{n_t}, \\ &\qquad \forall i,j: \ \mathbf{Q}_{ij} \in \{0,1\}, \quad \mathbf{Q}\mathbf{1}_{c_s} = \mathbf{1}_{c_t}, \end{aligned}$$

where $\| \cdot \|_F$ denote the Frobenius norm and $\mathbf{1}_d$ is a d-dimensional column vector of all ones. The constraints ensure that the elements of the matrix $\mathbf{P}$ are binary-valued and that each target example is mapped to exactly one source example. The first term in the objective function is a measure of reconstruction error when mapping the target examples into instances of the source domain. The second term in the objective function ensures consistency of the class labels, i.e., labeled examples of a particular class in the target domain are mapped only to source examples of the same class. The label matching matrix $\mathbf{Q}$ is a $c_s \times c_t$ binary-valued matrix that represents the mapping between the class labels of the source and target domains. Assuming $c_s = c_t = c$ and since the classes in the source and target domains are often unrelated, we found it is sufficient to assign $\mathbf{Q}$ to be an identity matrix when performing our experiments. A more careful selection of $\mathbf{Q}$ would require considerations of the within-class and between-class variability of the source and target examples. We plan to pursue this as part of our future work.

The constraint on matrix $\mathbf{P}$ allows a many-to-one assignment between the target and the source domains. This is essential because, for any two target examples that are located close to each other and belong to same class, the surrogate selection matrix should map them to

source examples that are close to each other as well. If no such corresponding pair of source examples can be found, it would be better to map the two target examples to the same surrogate, as long as they are both from the same class.

6.4.1 Reconstruction Error Analysis

Our proposed framework considers a linear transformation approach for mapping the target examples into their corresponding surrogates in the source domain. In the case where the source domain corresponds to an image corpus, one concern is whether the transformation can produce images that can be easily discerned by humans. One way to measure the quality of the transformation is to evaluate the reconstruction error of the surrogates selected for the target examples. Let $\hat{\mathbf{X}}^{(s)} = \mathbf{X}^{(tl)}\mathbf{U}$ be the transformed images of the target examples and $\mathbf{Z}^{(s)} = \mathbf{P}\mathbf{X}^{(s)}$ be the selected surrogates. Given $\mathbf{U}$ and $\mathbf{P}$, we consider the transformation to be ϵ-proper if the reconstruction error $\|\mathbf{Z}^{(s)} - \hat{\mathbf{X}}^{(s)}\|_F^2 \leq \epsilon$ and exact if $\|\mathbf{Z}^{(s)} - \hat{\mathbf{X}}^{(s)}\|_F^2 = 0$. The transformed images of the target examples are expected to be as easily discernable as the original source images themselves if ϵ is small. A key question is whether it is possible to construct a transformation matrix $\mathbf{U}$ with low reconstruction error given the source and target data matrices $\mathbf{X}^{(s)}$ and $\mathbf{X}^{(tl)}$.

To determine the condition under which a low reconstruction error can be obtained, assume the surrogate selection matrix $\mathbf{P}$ is known. The reconstruction error can be written as follows

$$\|\mathbf{Z} - \mathbf{X}^{(tl)}\mathbf{U}\|_F^2 \tag{6.1}$$

whose minimum solution is given by

$$\mathbf{U} = \left(\mathbf{X}^{(tl)T}\mathbf{X}^{(tl)}\right)^{-1}\mathbf{X}^{(tl)T}\mathbf{Z} \tag{6.2}$$

A unique solution exists only if the covariance matrix $\mathbf{X}^{(tl)T}\mathbf{X}^{(tl)}$ is of full column rank. Otherwise we need to obtain a rank reduced approximation of $\mathbf{X}^{(tl)}$ using techniques such as singular value decomposition (SVD).

More importantly, the reconstruction error of the transformation can be assessed in terms of the rank of the data matrices.

Proposition 1 *Let* $\mathbf{A}$ *be an* $m \times n$ *matrix and* $\mathbf{B}$ *be an* $n \times k$ *matrix. If* $r(\mathbf{A})$*,* $r(\mathbf{B})$*, and* $r(\mathbf{AB})$ *denote the ranks of matrices* $\mathbf{A}$*,* $\mathbf{B}$*, and* $\mathbf{AB}$*, respectively, then it can be shown that [79],*

$$r(\mathbf{AB}) \leq \min[r(\mathbf{A}), r(\mathbf{B})] \tag{6.3}$$

Since $\hat{\mathbf{X}}^{(s)} = \mathbf{X}^{(tl)}\mathbf{U}$, according to this proposition, $r(\hat{\mathbf{X}}^{(s)}) \leq r(\mathbf{X}^{(tl)})$. Thus, if the rank of the target data matrix is considerably lower than that for the original source data matrix, then the transformed images will have a lower rank than the original source images, which in turn, may lead to large reconstruction errors. In other words, an exact or low ϵ-proper transformation is infeasible if $r(\mathbf{X}^{(tl)}) \ll r(\mathbf{X}^{(s)})$.

We illustrate this with an example in Figure 6.2. Here we consider the well known Iris data consisting of 150 labeled examples belonging to 3 distinct categories (Iris versicolor, Iris virginica, and Iris setosa) as our target domain. Each category contains 50 examples, which are matched against 50 handwritten images of 28×28 dimensions containing the digits 1, 2, or 3. The rank of the data matrix for the handwritten images is 150 (which is equivalent

to its number of rows), which is much higher than the rank of Iris data, which is equal to 4 (i.e., its number of columns). According to Proposition 1, the rank of the transformed images for the Iris data is at most 4, which is considerably lower than the rank of the original handwritten images. This suggests that the reconstruction error for the Iris data using the handwritten images is likely to be high. The top right column of the Figure 6.2 shows the transformed images for 30 selected examples from the Iris data. Even though all the target examples in the Iris versicolor class were mapped to surrogate images containing the digit 2, the transformed images look noisy and do not resemble the digit 2. Instead, they looked like a mixture of digits 2 and 3 because it is hard to distinguish target examples belonging to the Iris versicolor class from those belonging to the Iris virginica class. However, if we increase the dimensionality of the Iris data from 4 to 12 (by adding quadratic and cubic terms for each of the 4 original features), the reconstruction error reduces significantly, especially for those images that correspond to the Iris versicolor class (see bottom right column of Figure 6.2). In particular, if we project the target data to a 150-dimensional feature space (using higher degree polynomials), an exact transformation matrix $\mathbf{U}$ can be obtained.

Proposition 2 *If* $\mathbf{Z}^{(s)} \in \Re^{n_s \times d_s}$, $\mathbf{X}^{(tl)} \in \Re^{n_t \times d_t}$, *and* $r(\mathbf{X}^{(tl)}) \geq r(\mathbf{X}^{(s)})$ *(where* $\mathbf{X}^{(tl)}$ *is a full column rank matrix), then there exists a transformation matrix* $\mathbf{U}$ *such that* $\mathbf{Z}^{(s)} = \mathbf{X}^{(tl)}\mathbf{U}$.

The key lesson here is that it is preferable to have a source data whose rank is smaller than that of the target data. There are two ways to achieve this. First, we can reduce the rank of the source data by applying SVD. The drawback here is that the source data is typically an image whose contents are manually evaluated by humans (crowds). The rank reduction may damage the visual clarity of the source images. Alternatively, we can increase the rank of

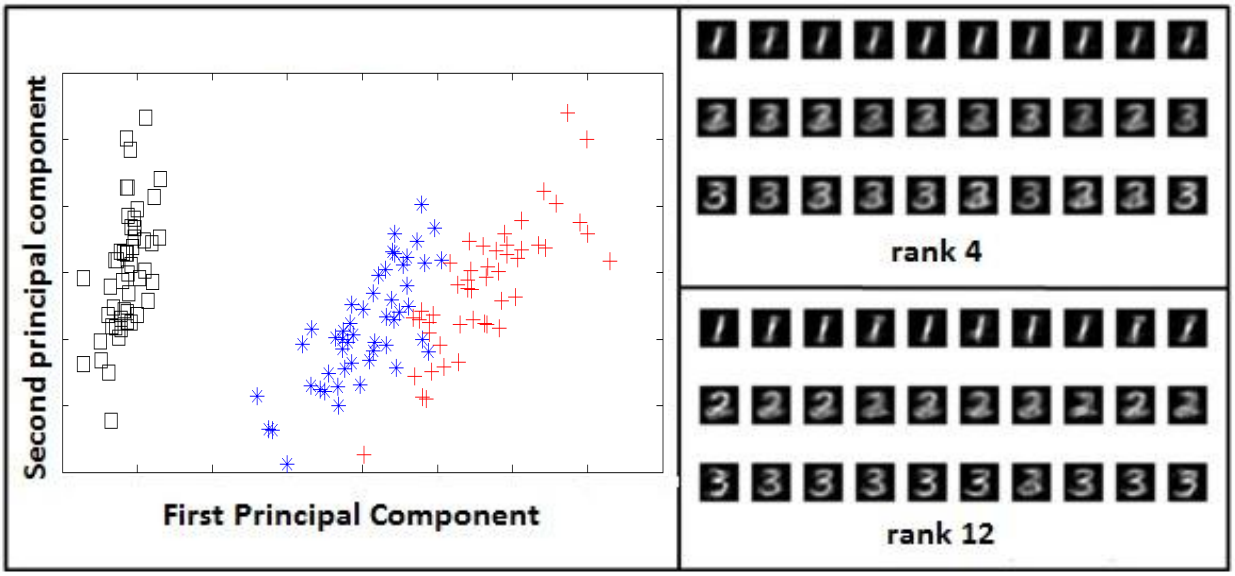

FIGURE 6.2: Left panel shows the distribution of three classes (in three colors) with respect to first and second principal components. The top and bottom right panel gives the transformed data using 4 and 12 features respectively. The blurry images are formed because of poor transformation quality. This is discussed in detail in section 6.4.1

the target data by adding features that correspond to higher order polynomials of the target attributes. In fact, the objective function of our proposed surrogate matching framework can be extended to a nonlinear transformation using the kernel trick (see supplemental material for derivation). This allows us to project the target data to a high-dimensional (possibly infinite-dimensional) space, thereby making it compatible with the source data of higher rank.

6.4.2 Parameter Estimation

This section presents our approach for estimating the parameters $\mathbf{P}$ and $\mathbf{U}$ for the constrained optimization problem stated in Section 6.4. We also provide proof of convergence of the proposed algorithm. First, note that the objective function is non-convex with respect to both $\mathbf{P}$ and $\mathbf{U}$. However, for a fixed $\mathbf{P}$, it is convex with respect to minimizing $\mathbf{U}$, and vice-versa. Thus, we employ the well-known alternating least square method to solve the

optimization problem.

We begin with an initial surrogate selection matrix $\mathbf{P}^0$ satisfying the label compatibility criterion, namely $\mathbf{P}^0 Y_s \mathbf{Q} = Y_l$ (assuming $\mathbf{Q} = \mathbf{I}$ as previously discussed). We then estimate the transformation matrix $\mathbf{U}^0$ using (6.2). At iteration k, we estimate $\mathbf{P}^k$ based on the previous estimate for $\mathbf{U}^{k-1}$ as follows. For each target example $\mathbf{x}_i^{(tl)}$, we compute its transformed image $\hat{\mathbf{x}}_i^{(tl)} = \mathbf{x}_i^{(tl)T} \mathbf{U}^{k-1}$ and match it to the surrogate example $\mathbf{x}_{k_i}^{(s)}$ that minimizes its reconstruction error:

$$k_i = \text{argmin}_{j: y_j^{(s)} = y_i^{(tl)}} \parallel \mathbf{x}_j^{(s)} - \hat{\mathbf{x}}_i^{(tl)} \parallel^2 \tag{6.4}$$

We then set $\mathbf{P}_{i,k_i}^k = 1$ and $\mathbf{P}_{i,j}^k = 0$ $(\forall j \neq k_i)$. The condition $y_j^{(s)} = y_i^{(tl)}$ imposes the label compatibility requirement on the selected surrogate.

Lemma 1 *Let $\boldsymbol{X}^{(tl)}$ be a full column rank matrix. Then, the objective function* (6.1) *is monotonically non-increasing as the number of iterations k increases in Algorithm 3.*

Proof 1 *To prove the result, we need to show the following inequality:*

$$\mathcal{E}[\boldsymbol{P}^k, \boldsymbol{U}^k] \leq \mathcal{E}[\boldsymbol{P}^k, \boldsymbol{U}^{k-1}] \leq \mathcal{E}[\boldsymbol{P}^{k-1}, \boldsymbol{U}^{k-1}], \tag{6.5}$$

where $\mathcal{E}[\boldsymbol{P}^k, \boldsymbol{U}^k]$ is the reconstruction error after the k-th iteration. Given $\boldsymbol{U}^{k-1}$, the new selection matrix $\boldsymbol{P}^k$ satisfies the condition that $\forall i, \quad \boldsymbol{P}_{i,k_i}^k = 1$ implies,

$$\|\mathbf{x}_{k_i}^s - \mathbf{x}_i^{(tl)^T} \boldsymbol{U}^{k-1}\| \leq \ \|\mathbf{x}_j^s - \mathbf{x}_i^{(tl)^T} \boldsymbol{U}^{k-1}\| \ : \forall j \neq k_i \tag{6.6}$$

The preceding inequality follows from the algorithm step given in Equation (6.4)*. Now sup-*

Algorithm 3 `Surrogate Mapping Algorithm`

Input: Source and Target data $\mathbf{X}_m$, $\mathbf{X}_l$
Output: Transformation Matrix $\mathbf{U}$
Initialize:
  k = 0;
  Initialize $\mathbf{P}^0$, satisfying $\mathbf{P}^0 Y_s = Y_l$

repeat
    $k = k + 1$
    Learn the transformation $\mathbf{U}^{k-1}$
        $\mathbf{U}^{k-1} = \text{argmin}_{\mathbf{U}} \parallel \mathbf{P}^{k-1}\mathbf{X}^{(s)} - \mathbf{X}^{(tl)}\mathbf{U} \parallel^2$
    Compute the transformed image as follows
        $\hat{\mathbf{x}}_i^{(tl)} = \mathbf{x}_i^{(tl)T}\mathbf{U}^{k-1} \qquad \forall i = 1, 2, \ldots, n$
    Estimate the $\mathbf{P}^k$ matrix
        Set $\mathbf{P}^k$ =0
    for $i = 1 \rightarrow n$ **do**
            $k_i = \text{argmin}_{j:y_j^{(s)}=y_i^{(tl)}} \parallel x_j^s - \hat{x}_i \parallel^2$
        $\mathbf{P}^k_{i,k_i} = 1$
    end for
until $\mathbf{P}^{k-1} = \mathbf{P}^k$
`return` $\mathbf{U}^{k-1}$

pose we assume that $\mathcal{E}[\boldsymbol{P}^k, \boldsymbol{U}^{k-1}] > \mathcal{E}[\boldsymbol{P}^{k-1}, \boldsymbol{U}^{k-1}]$. *Then by definition of re-construction error given in* (6.1),

$$\begin{aligned} & \|\mathbf{P}^k\mathbf{X}^{(s)} - \mathbf{X}^{(tl)}\mathbf{U}^{k-1}\|_F^2 > \|\mathbf{P}^{k-1}\mathbf{X}^{(s)} - \mathbf{X}^{(tl)}\mathbf{U}^{k-1}\|_F^2 \\ \Rightarrow \quad & \sum_i [\mathbf{x}^s_{k_i} - \mathbf{x}_i^{(tl)T}\,\boldsymbol{U}^{k-1}]^2 > \sum_i [\mathbf{x}^s_{(k-1)_i} - \mathbf{x}_i^{(tl)T}\,\boldsymbol{U}^{k-1}]^2 \end{aligned} \tag{6.7}$$

Since each term in the sum is non-negative, there must exist an index i such that

$$[\mathbf{x}^s_{k_i} - \mathbf{x}_i^{(tl)}\,\boldsymbol{U}^{k-1}]^2 > [\mathbf{x}^s_{(k-1)_i} - \mathbf{x}_i^{(tl)}\,\boldsymbol{U}^{k-1}]^2 \tag{6.8}$$

which contradicts the condition given in (6.6). *Thus, the original assumption must be false, which means* $\mathcal{E}[\boldsymbol{P}^k, \boldsymbol{U}^{k-1}] \leq \mathcal{E}[\boldsymbol{P}^{k-1}, \boldsymbol{U}^{k-1}]$.

The next inequality $\mathcal{E}[\boldsymbol{P}^t, \boldsymbol{U}^t] \leq \mathcal{E}[\boldsymbol{P}^t, \boldsymbol{U}^{t-1}]$ *can be inferred from line* 10 *of Algorithm 3 (see Equation* (6.2)*). Since* $\boldsymbol{X}^{(tl)}$ *is a full rank matrix, therefore* $\boldsymbol{U}^k$ *must exist for every* k *and is unique.*

Theorem 3 *Let* $\boldsymbol{X}^{(tl)}$ *be a full column rank matrix. Then the Algorithm 3 terminates after finitely number of iterations.*

Proof 2 *Lemma 1 shows the monotonically non-increasing nature of the objective function* (6.1) *when applying Algorithm 3. At each iteration, the algorithm re-estimates the selection matrix* $\boldsymbol{P}$ *and subsequently finds a corresponding optimal transformation matrix* $\boldsymbol{U}$*. Since there are only finitely many permutations available for matrix* $\boldsymbol{P}$*, in the worse-case scenario, the algorithm should converge after considering all the permutations.*

We need to show is that the Algorithm does not enter infinite loop. For this, it is sufficient to show that the algorithm will not produce the same selection matrix $\mathbf{P}$ *again and again except in the last iteration. If this is not true, then let* $\boldsymbol{P}^t, \boldsymbol{U}^t$ *be the permutation matrix and transformation matrix obtained at the end of iteration* t *and let* $\boldsymbol{P}^{t+1}$ *be the new permutation matrix estimated using* $\boldsymbol{U}^t$ *such that* $\boldsymbol{P}^{t+1} = \boldsymbol{P}^i$ *for some* $i < t$ *and* $i \neq t$*. Since the linear transformation* $\boldsymbol{U}$ *is unique for a given permutation matrix, we have*

$$\begin{aligned} \boldsymbol{U}^{t+1} &= argmin_{\boldsymbol{U}} \parallel \boldsymbol{P}^{t+1}\boldsymbol{X}^{(s)} - \boldsymbol{X}^{(tl)}\boldsymbol{U} \parallel^2 \\ &= argmin_{\boldsymbol{U}} \parallel \boldsymbol{P}^{i}\boldsymbol{X}^{(s)} - \boldsymbol{X}^{(tl)}\boldsymbol{U} \parallel^2 \\ &= \boldsymbol{U}^i \end{aligned} \tag{6.9}$$

However, this results in following contradiction.

$$\mathcal{E}[\boldsymbol{P}^i\,\boldsymbol{U}^i] < \mathcal{E}[\boldsymbol{P}^t, \boldsymbol{U}^t] \leq \mathcal{E}[\boldsymbol{P}^{t+1}, \boldsymbol{U}^{t+1}] = \mathcal{E}[\boldsymbol{P}^i\,\boldsymbol{U}^i] \tag{6.10}$$

Therefore, our assumption $i \neq t$ is wrong. In other words $\boldsymbol{P}^{t+1} = \boldsymbol{P}^{i}$ can only happen when $i = t$ or in the last iteration, at which the algorithm is said to have terminated.

6.5 Experimental Evaluations

We performed experiments using both synthetic and real world data to demonstrate the effectiveness of the proposed data transformation technique in enabling the use of crowdsourcing technology for labeling the network data. For this purpose, we have selected the grey scale image corpus of handwritten digits [51], as source data domain. This source domain consists of roughly 5000 images for each of the digits from 0 through 9. Each image is of size 28×28 and is represented as one dimensional vector of length 784. We used two sparse data as target domains namely, a social network of Wikipedia editors and a sample collection of Wikipedia articles. A detailed description of these two data sets are given below.

- **Wikipedia Editor Interactions:** Here, we sampled articles from Wikipedia on two topics namely, computer science and natural science. We then took all the editors who worked on at least 20 of the sampled articles and recorded the interactions between them from their respective Wikipedia User::Talk pages. We created two editor networks one for each of the two topics. In each network, the set of editor interactions (linked node pairs) and non-interactions(non-link node pairs) were equi-partitioned to create TRAIN and TEST set. Therefore, each unique editor-pair is present in either TRAIN or TEST set. The goal is to learn the presence/absence of interactions between editor pairs from the TRAIN set and predict the interactions between the editors on the TEST set.

- **Biology Article Corpus:** We sampled Wikipedia articles belonging to 4 related

topics in biology namely - genetics, zoology, anatomy and cell-biology. Here, instead of the network link information, we have used the text data (words in the article) to classify the articles into one of the four categories. The TRAIN set consists of 2000 articles with 500 article in each topic and TEST set consists of 800 articles with 200 articles from each topic. There are totally 6040 words in the TRAIN set, which were used as features. The goal is to obtain correct labels for the TEST articles, without exposing the article content to the crowd.

6.5.1 Evaluation Methodology and Baseline

Our goal is to learn the labels for the TEST data points as accurately as possible. As a baseline, we have trained a support vector machine (SVM) on the TRAIN set and used it to predict the labels for the points in the TEST set. In the proposed approach, we first learn the transformation matrix $\mathbf{U}$ between the labeled TRAIN set and the source domain (set of digit images). We then apply the transformation on the unlabeled points in the TEST set. The transformed TEST data now resides in the source domain (as handwritten digit images). We have utilized the services of crowd(non-domain expert human workers) to manually label these surrogate digit images (transformed test data) into appropriate digit category. If the image is not well formed or appears visually cluttered then the worker flags the data point (image) as noise. We combine the evaluations of all the workers (individual members in crowd) by taking a simple majority vote.

The crowd based approach has the flexibility of rejecting a data point as noise. This approach uniquely partitions the TEST set into two groups viz. good images (TEST_G) and noisy images (TEST_N). Their performance can only be reported on the respective TEST_G set. Therefore, it is not fair to compare their performance against the baseline. In order

to address this problem, we performed additional model building step that would generate label for the entire TEST data. It is a semi supervised approach where we trained a SVM on both TRAIN and TEST_G set and used it to predict labels for entire TEST set. TEST_G is subset of TEST where every data point is assigned a unique label by the crowd (majority vote). We denote this approach by `Crowd + SVM`.

6.5.2 Synthetic Data

We used synthetic data to understand the ability of the proposed out-of-domain mapping algorithm in revealing the latent structure present in the target data. In particular, we examined the ability of the proposed approach in distinguishing boundary points from non boundary points in a simple binary classification problem. In order to study this, we have generated 100 random samples each from two normal distributions namely $\mathcal{N}(\mu = 1, \sigma = 1)$ and $\mathcal{N}(\mu = 3, \sigma = 1)$. We converted the one dimensional data to 50 dimension data by appending polynomials of the feature from degree 1 through 50. Samples from the former are mapped to images corresponding to digit 0 and samples from the later are mapped to images corresponding to digit 1.

Figure 6.3 depicts the two normal distribution from which we sampled 200 points. Figure 6.4 shows the transformed data (image representation) for all the 200 sample points. The target domain consists of real numbers from the interval $[-3 \;\; 5]$. For individual data point, the actual sample value is printed on top of each image. The data points belonging to class 0 ($\mathcal{N}(\mu = 1, \sigma = 1)$) are marked by $*$ sign. For the samples from this distribution, all the sample points in the interval $[-2 \;\; 1.5]$ got mapped to digit image zero without any clutter or noise. Similarly, for the samples from class 1 ($\mathcal{N}(\mu = 3, \sigma = 1)$), all the sample points in the interval $[2.5 \;\; 5]$ got mapped to the digit image 1 without any noise. However,

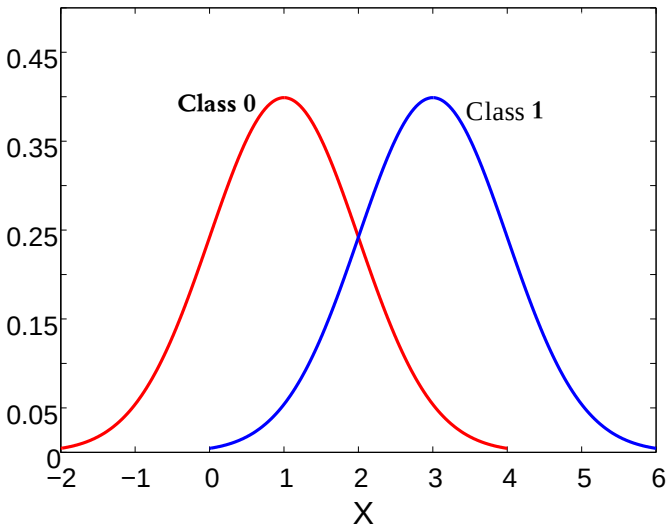

FIGURE 6.3: Plot of two normal distributions used in generating synthetic data.

the sample points from the interval [1.5 2.4] got mapped to image which contains both digit 0 and digit 1. In fact, this interval marks the overlapping region or boundary region between the two distributions and the transformed target images for points from this region looks like alphabet Φ. This effect remains same for higher data dimensions where the visual representation of data points in the boundary region encapsulates the characteristics of all the respective classes that share the boundary. This is further illustrated in the next section on a real world example.

6.5.3 Biology Article Corpus

On this data corpus, we have experimentally validated the Theorem 3 and other propositions. Here we have mapped the articles belonging to four categories namely, genetics, zoology, anatomy and cell-biology to digit images $1, 2, 3$ and 4 respectively. There are 500 articles in each article category and we have randomly matched each article against a unique image from appropriate digit category. This is done by the initial permutation matrix $\mathbf{P}^0$. Since, the final surrogate mapping depends on the initial permutation matrix $\mathbf{P}^0$, we performed

FIGURE 6.4: The transformed Data from two normal distributions. The text above each image is not meant to be readable but for visual reference only. The number above each image is the actual sample value. The interpretation of images is illustrated in section 6.5.2.

the experiment for 10 different random initializations of $\mathbf{P}^0$ and chose the transformation that gave the minimum reconstruction error.

On this data set, we have experimentally validated the Propositions, Lemma and theorem proved in this chapter. In order to validate Lemma 1, we plot the reconstruction error obtained at each iteration. This is shown in Figure 6.5(left) for each of the 10 random initializations. As proved in Lemma 1, the reconstruction error monotonically decreases with each iteration for all the 10 random initializations of $\mathbf{P}^0$. It was also observed that for each random initialization, the reconstruction error converged to a local minima albeit different ones for each random initialization thus validating Theorem 3. As mentioned earlier, the re-construction can be analyzed with respect to the rank of the source and the target data matrices. Figure 6.5(right), shows the decreasing trend of the rank of the re-sampled source matrix with each iteration. It should be noted that the rank of $\mathbf{P}^t\mathbf{X}_s$ does not always decrease monotonically with each iteration. The overall decrease in rank can be attributed to the fact that at each iteration, the algorithm performs a one-to-many mapping, effectively reducing the row rank of the matrix $\mathbf{P}^t\mathbf{X}_s$. Notice that the minimum error across all the 10 random runs is obtained when the rank of $\mathbf{P}^T\mathbf{X}_s$ falls below the rank of target data. However, that alone is not sufficient condition for decreasing the re-construction error. In Figure 6.5, errorcurve denoted by legends $\circ, \lozenge$ and $\square$ correspond to the condition where the rank of $\mathbf{P}^t\mathbf{X}_s$has fallen below rank of the target data.

The surrogate images generated by the transformation with minimum re-construction error is then presented to a crowd of three workers. On this dataset, each of the three workers manually labeled all the 800 TEST images. Each worker categorized an image into one of the four digit $(1-4)$ category. If the surrogate image was cluttered, dark or contained more than one digit, then it was labeled as noise. On an average, each worker has labeled

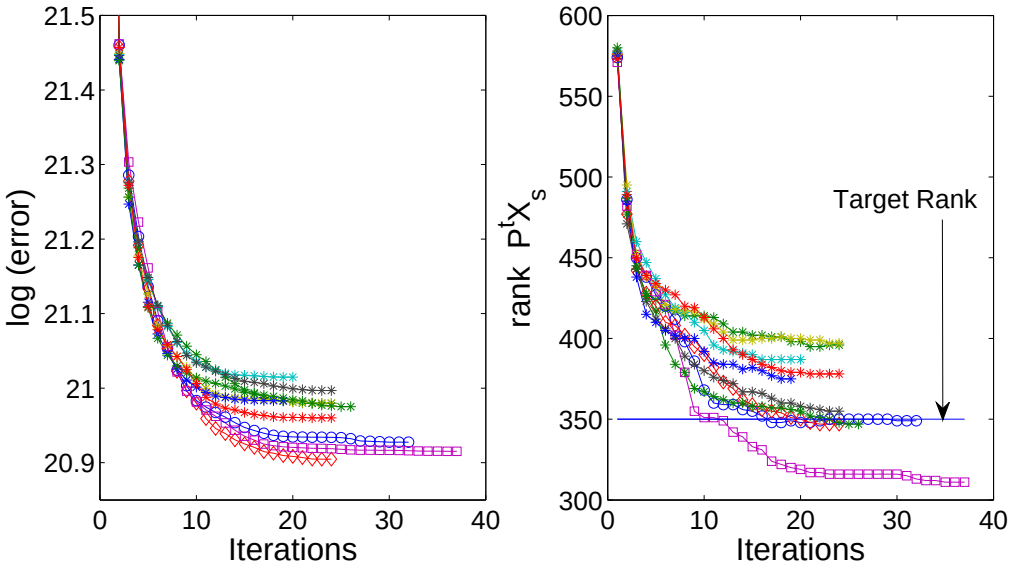

FIGURE 6.5: **Wikipedia article corpus:** The left result shows the decreasing error with each iteration for 10 random initializations. The right figure shows the rank of the source data almost decreases with successive iterations. Lowest error is achieved when the rank of the source falls below the target rank.

90 TEST data points or 11% of the TEST set as noise. The classification accuracy of three independent workers on the entire TEST set was found to be $74\%, 72\%$ and 73%. For the baseline, we trained a SVM on the TRAIN set using radial basis kernel with sigma value of 0.1. The SVM gave an over all accuracy of 63.5% on the TEST set. Table 6.1 records the F measure for each class as given by SVM and each of the three individual workers in the crowd.

An important distinguishing aspect of the crowdsourcing approach from the SVM is that the latter was directly trained and tested on target domain whereas in the former, workers manually assign label by looking at the transformed data (digit images) residing on the source domain. In addition, the performance of the SVM directly depends on the data points in the sample TRAIN set, the choice of kernel and kernel parameters. On contrary, the performance of the crowd depends on the quality of the transformation, the user interface

TABLE 6.1: This table shows the F measure for each of the four article categories. Here W_i denotes the i^{th} worker in the crowd. Each independent worker performs better than SVM with rbf kernel ($\sigma = 0.1$). The `Crowd + SVM` performed better than individual workers in the crowd.

Class	SVM	W1	W2	W3	`Crowd+SVM`
	AUC				
Zoology	0.5639	0.6977	0.6667	0.7178	0.7079
Cell Biology	0.7250	0.8677	0.8571	0.8673	0.8756
Anatomy	0.6267	0.7436	0.7512	0.7350	0.7400
Genetics	0.6285	0.7866	0.7798	0.7803	0.8104
Accuracy	63.50	74.75 %	72.12 %	72.88%	78.25%

for labeling, worker skills and fatigue. In this experiment, the workers were not given any special training or instruction for labeling. They were not exposed to the images formed on the TRAIN set and they used the basic human intelligence in categorizing the images into one of the four digits or noise.

Finally in order to obtain a unique label for each TEST data point, we have used a simple majority vote strategy to combine the results of individual workers in the crowd. Here a TEST data point is assigned to a category which was favored by majority of workers in the crowd (atleast two out of three workers). This approach labels only a portion of TEST set. We generalize this crowd generated label on entire TEST set as follows. We train a SVM on the combined TRAIN set and crowd labeled portion of TEST set and use it to predict the label for entire TEST set. This approach is denoted as `Crowd + SVM` and it gave an accuracy of 78.25 as reported in the Table 6.1.

6.5.4 Wiki Editor Networks

In this section, we analyze the performance of the proposed approach for label acquisition on the Wikipedia editor networks. Each Wikipedia article is composed and edited by several editors and each editor works on several articles. Sometimes, the editors interact with each

other through their respective USER::Talk pages to improve the content of articles. In our sample, we observed that a large pool of editors have worked on a common set of articles without interacting with each other. Therefore merely editing a common set of article does not imply interaction. For each editor, we use a binary vector indicating whether or not the editor worked on a article as attribute vector. For each editor pair, the sum of their corresponding attributes is used as feature vector. Here again, we used the SVM as baseline algorithm to predict the interactions between the editors. We trained the SVM using linear kernel and rbf kernels with different σ parameters. The best result was obtained with the linear kernel.

The editor networks data is similar to the Biology article corpus data in that both are sparse, high dimensional data and require strong domain expertise to label them. However learning on the editor network data poses a significant challenge owning to the label skewness. In our sample, the number of non-interacting editors are $2 - 10$ times than the number of interacting editor pairs. In addition, the attributes or features used for predicting interactions has limited discerning capability. This results in generating images containing multiple digits or undecipherable characters which cannot be easily labeled by humans.

Another distinguishing aspect between the experiments on Biology article corpus and Editor network is that on the former there were three workers who labeled the entire TEST set. However, on the later there were 27 workers who labeled only a portion of TEST set (on each network, 800 TEST images were labeled). Therefore we do report the performance of individual workers rather, we report the overall performance of the `Crowd+SVM` approach.

We mapped the data points corresponding to the interactions(links) to digit 1 and the data belonging to non-interaction(no-links) group were mapped to digit 2. For each surrogate image, a worker chose one of the following five options: A) Clearly digit 1 B) Clearly digit

TABLE 6.2: This table gives the AUC values of SVM and `Crowd + SVM` approach for link prediction problem on two different Wikipedia editor networks. Here the proposed `Crowd` approach outperforms the baseline SVM by 8% on Computer science network and fares slightly better than SVM on Natural science editor network.

	AUC		
Network	Pos/Neg in TEST	SVM	`Crowd + SVM`
Nat.Sc.	489/1003	0.7507	0.7701
Comp. Sc.	155/1500	0.7287	0.7950

2, C)Prominent 1, with little 2, D) Prominent 2 with little 1 and E) Noise. The surrogate images assigned to category A and B were assigned label $+1$ and -1 respectively. The surrogate images assigned to category C and D were assigned label 0.5 and -0.5 respectively. The images assigned to category E were discarded as noise. We then generalized the label over entire TEST set by training SVM on the TRAIN and TEST_G. On an average only 130 images (roughly 10%) of the TEST images in each network were assigned to single category by the crowd. In Table 6.2, we report the area under the curve (AUC) values for SVM and `Crowd` on each of the two networks. The SVM model trained on the TRAIN set gave an AUC of 75% and 72% respectively on the natural science and computer science editor networks. However, the `Crowd + SVM` approach further lifted the AUC to 77% and 79% respectively. This shows that the partial label information generated by the crowd offers new insight to the SVM model to change the decision surface which inturn results in higher performance.

From the above experiments, we have demonstrated the capability of proposed framework in mapping the complex high dimensional sparse network data into a human comprehensible visual data that can be manually labeled by the crowd. The results suggest that learning approach on the surrogate data with assistance from the crowd outperforms (at the very least as good as) learning on the raw data. Throughout this work, we have used a simple majority vote among the workers to determine the best label for each surrogate image. Sophisticated

algorithms [37, 45, 113] can be used to combine the label information from different workers to generate a final reliable label for the given surrogate image and the corresponding network data point.

6.6 Conclusions

In this chapter, we propose a transformation learning technique that learns a point transformation between any two data domains. We highlight the challenges associated with this problem and analyze the data requirements for learning the point mapping. We list two application of such transformation namely data obfuscation for performing privacy preserving data mining and visual representation of data. In this work, we apply the proposed technique to transform sparse, high dimensional social network data into a set of digit images that are labeled by a non domain experts. This way we perform network mining tasks like link prediction and node classification using crowd-sourcing technology.

Chapter 7

Future Work

In this thesis, we have presented three learning frameworks for combining data from multiple sources to perform common network mining tasks. Firstly, we have presented a framework to jointly perform clustering and/or classification on two related network domains. Secondly, we have presented a framework to simultaneously perform related learning tasks namely, community detection and link prediction. Finally, we have presented a generic framework to leverage label information from unrelated non-network data domain to perform supervised learning on networks. We have demonstrated the performance of all the proposed framework on real world networks like citation network, co-authorship network and user networks. In this chapter, we highlight the directions for future research on two novel problems that are unique to this thesis, namely, the multi-task learning on networks and out-of-domain transformation for label augmentation.

1. **Multi-task Learning:** In Chapter 5, we have presented a unique loss function for jointly performing the link prediction and community detection tasks on the network data. We have showed the relationship between the proposed loss function and the well known modularity measure for community detection. We demonstrated its applicability on a single network and in future, this could be extended to multiple related networks as well. To give an example, the framework can be employed to predict the missing citation links between relevant Wikipedia articles and at the same time identify

communities between corresponding authors or editors. In other words, the framework can be easily extended to perform link prediction on one network and community detection on another network.

Another important problem in network mining is identification of influential nodes in the network. These are minimum set of nodes that influences the diffusion of information in a network. Clearly, each community has its own set of influential nodes [4]. Once the communities are deciphered, the search for influential nodes can be narrowed to high degree nodes inside communities and bridge nodes between communities. Conversely, knowing the influential nodes in the network could expedite the process of community detection in a network, as each community has its own popular or influential nodes [107]. Therefore it is beneficial to perform these two tasks jointly. The proposed degree dependent loss function can be suitably altered to accomplish this task.

2. **Surrogate Mapping:** In Chapter 6, we have presented a novel data transformation framework in order to augment the label information for the given network mining task. This framework allows us to transform a large, sparse, high dimensional network data into a image format. We sincerely hope that the proposed approach of representing network data in a visual format for crowd labeling would be a trend setter for future research in this field. This offers several different directions for future research as listed below

    - Extend the proposed linear framework for surrogate mapping to a non-linear framework in order to minimize the transformation error. In addition, algorithmic approach should be designed to automatically perform label matching (estimating

Q matrix in equation (6.1)) between the domains.

- Currently, we use a simple majority voting to decide the final label for data points labeled by the crowd. However, sophisticated learning algorithms like boosting can be utilized to combine the labels generated by the individual workers in order to enhance the overall crowd performance.
- In this thesis, we have demonstrated the usefulness of the surrogate mapping algorithm for solving link prediction problem. It can be extended to solve multi-task learning problems on related networks like the ones we described in Chapter 3 and 4.

APPENDIX

Appendix

Proof of Convergence

In this section we provide the formal proofs that show the update formula (3.5) - (3.9) monotonically decrease the objective function (3.3). The proofs shown here has been reproduced from our previous work [20] of joint community detection across multiple networks.

Definition 2 *$G(w, \acute{w})$ is an auxiliary function for F(w) if following two conditions are satisfied*

$$G(w, \acute{w}) \geq F(w), \qquad G(w, w) = F(w) \tag{A.1}$$

Lemma 2 *If $G_1(w, \acute{w})$ and $G_2(w, \acute{w})$ are auxiliary functions for $F_1(w)$ and $F_2(w)$ respectively, then $G = G_1 + G_2$ is the auxiliary function for $F = F_1 + F_2$. Further, F is non decreasing under the update formula*

$$w^{t+1} = arg \min_{w} G(w, w^t) \tag{A.2}$$

Proof 3 *The proof for G as an auxiliary function for F follows directly from definition.*

Below, we give the proof for second part.

$$\begin{aligned} F(w^{t+1}) &= F_1(w^{t+1}) + F_2(w^{t+1}) \\ &\leq G_1(w^{t+1}, w^t) + G_2(w^{t+1}, w^t) \\ &\leq G_1(w^t, w^t) + G_2(w^t, w^t) \\ &= F(w^t) \end{aligned}$$

The third line follows from the fact that w^{t+1} minimizes the auxiliary function G. Thus, $G(w^{t+1}, w^t) \leq G(w^t, w^t)$ and $F(w^{t+1}) \leq F(w^t)$, which completes the proof.

The Equation (3.3) involves a summation of three distance functions $D(\cdot\|\cdot)$, Lemma 2 suggests that it is sufficient to show that minimizing the auxiliary function that bounds (3.3) would decrease the overall objective function. First, we give the auxiliary function for the term $D(A\|XUX^T)$ and the auxiliary function for other terms can be similarly derived. The update formula for X which minimizes $D(A\|XUX^T)$ is given by

$$X_{ij} = X_{ij}\left(\frac{\sum_{a=1}^{N}\left(\frac{A_{ia}[XU^T]_{aj}}{[XUX^T]_{ia}} + \frac{A_{ai}[XU]_{aj}}{[XUX^T]_{ai}}\right)}{\left(\sum_{a=1}^{N}[XU + XU^T]_{aj}\right)}\right) \tag{A.3}$$

The objective function can be written as

$$F_1 = \sum_{i,j} A_{ij} \log \frac{A_{ij}}{\sum_{kl} X_{ik} U_{kl} X_{lj}^T} - A_{ij} + \sum_{kl} X_{ik} U_{kl} X_{lj}^T \tag{A.4}$$

Now define

$$\alpha_{k,l} = \frac{X_{ik}^{(t)} U_{kl} X_{lj}^{(t)T}}{\sum_{rs} X_{ir}^{(t)} U_{ks} X_{sj}^{(t)T}} \tag{A.5}$$

Clearly, $\sum_{rs} \alpha_{rs} = 1$. Using Jensens inequality we get

$$-\log \textstyle\sum_{kl} X_{ik} U_{kl} X^T_{lj} \leq -\sum_{kl} \alpha_{kl} \frac{X_{ik} U_{kl} X^T_{lj}}{\alpha_{kl}} \tag{A.6}$$

Substituting (A.6) in (A.4) we get $D(A \parallel XUX^T) \leq$

$$\sum_{ij} \Bigg[A_{ij} \log A_{ij} - A_{ij} - A_{ij} \sum_{kl} \alpha_{kl} \log \frac{X_{ik} U_{kl} X^T_{lj}}{\alpha_{kl}} + \sum_{kl} X_{ik} U_{kl} X^T_{lj} \Bigg] \tag{A.7}$$

plugging in α_{kl} in above equation gives the following bound on the objective function

$$\sum_{ij} \Bigg[A_{ij} \log A_{ij} - A_{ij} - A_{ij} \sum_{kl} \frac{X^{(t)}_{ik} U_{kl} X^{(t)T}_{lj}}{\sum_{rs} X^{(t)}_{ir} U_{rs} X^{(t)T}_{sj}} \Bigg(\log X_{ik} U_{kl} X^T_{lj} - \log \frac{X^{(t)}_{ik} U_{kl} X^{(t)T}_{lj}}{\sum_{rs} X^{(t)}_{ir} U_{rs} X^{(t)T}_{sj}} \Bigg) + \sum_{kl} X_{ik} U_{kl} X^T_{lj} \Bigg]$$

which is the auxiliary function for F_1. We denote it as $G_1(X, X^{(t)})$. Now taking derivative of G_1 with respect to X_{pq} we get,

$$\frac{\partial G}{\partial X_{pq}} = -\sum_{jl} A_{pj} \frac{X^{(t)}_{pq} U_{ql} X^{(t)T}_{lj}}{\sum_{rs} X^{(t)}_{pr} U_{rs} X^{(t)T}_{sj} X_{pq}} + \sum_{jl} U_{ql} X^{(t)T}_{lj} - \sum_{ik} A_{ip} \frac{X^{(t)}_{ik} U_{kq} X^{(t)T}_{qp}}{\sum_{rs} X^{(t)}_{ir} U_{rs} X^{(t)T}_{sp} X_{pq}} + \sum_{ik} X^{(t)}_{ik} U_{kq} = 0$$

We get

$$X_{pq} = X_{pq}^{(t)} \left[\frac{\sum_j \frac{A_{pj}[UX^{(t)T}]_{qj}}{[X^{(t)}UX^{(t)T}]_{pj}} + \sum_i \frac{A_{ip}[X^{(t)}U]_{iq}}{[X^{(t)}UX^{(t)T}]_{ip}}}{\sum_{j=1}^{N}[UX^{(t)T}]_{qj} + \sum_{i=1}^{N}[X^{(t)}U]_{iq}} \right] \tag{A.8}$$

which is same as (A.3). Similarly, the term $D(C \parallel XVY^T)$ can be written as

$$F_2 = \sum_{ij} C_{ij} \log \frac{C_{ij}}{\sum_{kl} X_{ik} V_{kl} Y_{lj}^T} - C_{ik} + \sum_{kl} X_{ik} V_{kl} Y_{lj}^T$$

Now define

$$\beta_{kl} = \frac{X_{ik}^{(t)} V_{kl} Y_{lj}^{(t)T}}{\sum_{rs} X_{ir}^{(t)} V_{rs} Y_{sj}^{(t)T}} \tag{A.9}$$

Once again $\sum_{kl} \beta_{kl} = 1$ and following the same procedure as above, we get the auxiliary function for $D(C \parallel XVY^T)$ to be $G_2(X, X^{(t)}) =$

$$\sum_{ij} \left[C_{ij} \log C_{ij} - C_{ij} - C_{ij} \sum_{kl} \frac{X_{ik}^{(t)} V_{kl} Y_{lj}^{(t)T}}{\sum_{rs} X_{ir}^{(t)} V_{rs} Y_{sj}^{(t)T}} \log X_{ik} V_{kl} Y_{lj}^T - \log \frac{X_{ik}^{(t)} V_{kl} Y_{lj}^{(t)T}}{\sum_{rs} X_{ir}^{(t)} V_{rs} Y_{sj}^{(t)T}} + \sum_{kl} X_{ik} V_{kl} Y_{lj}^T \right]$$

Taking derivative of this with respect to X_{pq} and equating it to zero, we get

$$X_{pq} = X_{pq}^{(t)} \left(\frac{\sum_{i=1}^{M} \frac{C_{pi}[YV^T]_{iq}}{[XVY^T]_{pi}}}{(\sum_{i=1}^{M} [YV^T]_{iq})} \right) \tag{A.10}$$

Minimizing the original objective function (3.3) with respect to X, we have

$$\begin{aligned} \min_X \mathcal{J}(X) &= \min_X F_1(X) + F_2(X) \\ &\leq \min_X G_1(X, X^{(t)}) + G_2(X, X^{(t)}) \end{aligned} \tag{A.11}$$

Taking derivative of $G_1(X, X^{(t)}) + G_2(X, X^{(t)})$ with respect to X and equating it to zero we get update formulae (3.5).

In all we have shown that $G_1(X, X^{(t)}) + G_2(X, X^{(t)})$ is auxiliary function for (3.3) with respect to variable X and (3.5) decreases this auxiliary function and hence the objective function. Therefore, we have proved Theorem 1

It should be noted that Theorem 1 states that the update formula monotonically decreases the objective function (3.3). However, it does not guarantee convergence to local minima or any stationary point. We stop the algorithm when the error between two consecutive iterations lies below a specified threshold or after certain maximum number of iterations.

The work by Finesso & Sperji [29], has shown the existence of solution for the problem $D(A \parallel WH)$. First, they argue that the minima exists in the interior of the domain. Then they show that by normalizing the A matrix (such that $\sum_{ij} A_{ij} = 1$ and adding stochastic constraint on H ($\sum_j H_{ij} = 1$) results in a update formula that converges to local minima in the interior of the domain (strictly positive quadrant).

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Z. Abraham and P.-N. Tan. A semi-supervised framework for simultaneous classification and regression of zero-inflated time series data with application to precipitation prediction. In *Proceedings of the 2009 IEEE International Conference on Data Mining Workshops*, ICDMW '09, pages 644–649, 2009.

[2] L. Adamic and E. Adar. Abstract how to search a social network, 2005.

[3] O. Alonso, C. Carson, D. Gerster, X. Ji, and S. U. Nabar. Detecting Uninteresting Content in Text Streams. In M. Lease, V. Carvalho, and E. Yilmaz, editors, *Proceedings of the ACM SIGIR 2010 Workshop on Crowdsourcing for Search Evaluation (CSE 2010)*, pages 39–42, Geneva, Switzerland, July 2010.

[4] M. Anjerani and A. Moeini. Selecting influential nodes for detected communities in real-world social networks. In *Electrical Engineering (ICEE), 2011 19th Iranian Conference on*, pages 1 –6, may 2011.

[5] L. Backstrom, D. P. Huttenlocher, J. M. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'06)*, pages 44–54, Philadelphia, PA, USA, August 2006.

[6] S. Banerjee, K. Ramanathan, and A. G. 0005. Clustering short texts using wikipedia. In *SIGIR*, pages 787–788, 2007.

[7] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. In *Science*, volume 286, 1999.

[8] A.-L. Barabsi and R. Albert. Emergence of scaling in random networks. *Science*, 286 (5439):509–512, 1999.

[9] S. Basu, A. Banjeree, E. Mooney, A. Banerjee, and R. J. Mooney. Active semi-supervision for pairwise constrained clustering. In *In Proceedings of the 2004 SIAM International Conference on Data Mining (SDM-04*, pages 333–344, 2004.

[10] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 59–68, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. doi: 10.1145/1014052.1014062. URL `http://doi.acm.org/10.1145/1014052.1014062`.

[11] M. Bilgic, G. Namata, and L. Getoor. Combining collective classification and link prediction. In *Proceedings of the ICDM Workshop on Mining Graphs and Complex Structures*, pages 381–386, Omaha, NE, USA, 2007.

[12] D. Cai, Z. Shao, X. He, X. Yan, and J. Han. Community mining from multi-relational networks. In *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, 2005.

[13] R. Caruana. Multitask learning. *Mach. Learn.*, 28:41–75, 1997.

[14] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. *SIGMOD International Conference on Management of Data*, pages 307–318, 1998.

[15] Chebotarev and S. E. V. The matrix-forest theorem and measuring relations in small social groups. In *Automation and Remote Control*, page 1505, 1997.

[16] F. Chen, P. N. Tan, and A. K. Jain. A co-classification framework for detecting web spam and spammers in social media web sites. In *CIKM*, 2009.

[17] A. Clauset, M. Newman, and C. Moore. Finding community structure in very large networks. In *Phys. Rev. E*, number 6, page 066111, 2004.

[18] A. Clauset, C. Moore, and M. Newman. Structural inference of hierarchies in networks. In *Intl. Conf. on Machine Learning, ICML*, 2006.

[19] M. Collins, R. E. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. *Machine Learning*, 48(1-3), 2002.

[20] P. M. Comar, P.-N. Tan, and A. Jain. Identifying cohesive subgroups and their correspondences in multiple related networks. In *Proc of the 2010 IEEE/WIC/ACM Int'l Conf on Web Intelligence (WI-2010)*, Toronto, Canada, 2010.

[21] P. M. Comar, P.-n. Tan, and A. K. Jain. Multi task learning on multiple related networks. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1737–1740, 2010.

[22] P. M. Comar, P.-N. Tan, and A. K. Jain. Linkboost: A novel cost-sensitive boosting framework for community-level network link prediction. In *ICDM*, pages 131–140, 2011.

[23] R. Das and M. Vukovic. Emerging theories and models of human computation systems: a brief survey. In *Proceedings of the 2nd international workshop on Ubiquitous crowdsouring*, UbiCrowd '11, pages 1–4, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0927-1. doi: 10.1145/2030100.2030102. URL `http://doi.acm.org/10.1145/2030100.2030102`.

[24] I. Davidson and S. S. Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *SDM*, 2005.

[25] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of the 9th ACM SIGKDD Int'l Conf on Knowledge Discovery and Data Mining*, pages 89–98. ACM, 2003.

[26] C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix tri-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD*, pages 126–135. ACM, 2006.

[27] E.Airoldi, D. Blei, S. Feinberg, A. Goldenberg, E. Xing, and A. Zheng. *Social Network Analysis: Models, Issues, and New Directions*, volume 4503 of *LNCS*. Springer, 2007.

[28] T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.*, 6:615–637, 2005.

[29] L. Finesso and P. Spreij. Nonnegative matrix factorization and i-divergence alternating minimization. *Linear Algebra and its Applications*, 416(23):270 – 287, 2006.

[30] G. Flake, K. Tsioutsiouliklis, and R. Tarjan. Graph clustering techniques based on minimum cut trees. *Technical Report*, 2002.

[31] G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 150–160. ACM Press, 2000.

[32] G. W. Flake, S. Lawrence, C. L. Giles, and F. M. Coetzee. Self-organization and identification of web communities. *IEEE Computer*, 35:66–71, 2002.

[33] L. Ford and D. Fulkerson. Maximal flow through a network. 8:399–404, 1956.

[34] B. Gallagher, H. Tong, T. Eliassi-Rad, and C. Faloutsos. Using ghost edges for classification in sparsely labeled networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 256–264, 2008.

[35] L. Getoor and C. P. Diehl. Link mining: A survey. *ACM SIGKDD Explorations*, 7(2): 3–12, December 2005.

[36] D. Gibson, J. M. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia: Links, Objects, Time and Space - Structure in Hypermedia Systems*, pages 225–234, Pittsburgh, PA, USA, June 1998.

[37] R. Gomes, P. Welinder, A. Krause, and P. Perona. Crowdclustering. In *NIPS*, pages 558–566, 2011.

[38] M. Grineva, M. Grinev, D. Turdakov, and P. Velikhov. Harnessing wikipedia for smart tags clustering. In *Proceedings of the International Workshop on Knowledge Acquisition from the Social Web (KASW2008)*, 2008.

[39] R. Guimera and M. Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. In *Proc. Natl. Acad. Sci. U.S.A.*, 2009.

[40] R. Guimerà, M. Sales-Pardo, and L. Amaral. Classes of complex networks defined by role-to-role connectivity profiles. *Nature Physics*, 2007.

[41] F. Harary. *Graph Theory*. Westview Press, 1994.

[42] M. Hasan, V. Chaoji, S. Salem, and M. J. Zaki. Link prediction using supervised learning. In *Workshop on Link analysis, Counter-terrorism, and Security*, Bethesda, MD, USA, April 2006.

[43] M. A. Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *In Proc. of SDM 06 workshop on Link Analysis, Counterterrorism and Security*, 2006.

[44] X. Hu, X. Zhang, C. Lu, E. K. Park, and X. Zhou. Exploiting wikipedia as external knowledge for document clustering. In *Proc. of Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2009.

[45] A. K. J. Jinfeng Yi, Rong Jin and S. Jain. Semi-Crowdsourced Clustering: Generalizing Crowd Labeling by Robust Distance Metric Learning. In *NIPS*, 2012.

[46] H. Kashima and N. Abe. A parameterized probabilistic model of network evolution for supervised link prediction. In *Proceedings of the ICDM*, pages 556–559, New York, NY, USA, 2006. ACM. doi: http://doi.acm.org/10.1145/956863.956972.

[47] H. Kashima and N. Abe. A parameterized probabilistic model of network evolution for supervised link prediction. In *Proceedings of the IEEE International Conference on Data Mining (ICDM 2006)*, 2006.

[48] H. Kashima, T. Kato, Y. Yamanishi, M. Sugiyama, and K. Tsuda. Link propagation: A fast semi-supervised learning algorithm for link prediction. In *Proceedings of the SIAM intl conf on Data Mining*, pages 1099–1110, Sparks, NV, USA, 2009.

[49] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, VOL. 18, NO. 1:39– 43, 1953.

[50] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[51] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[52] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, volume 13, pages 556–562, 2001.

[53] R. Lempel and S. Moran. Salsa: The stochastic approach for link-structure analysis. *ACM Transactions on Information Systems*, 2001.

[54] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 556–559, New York, NY, USA, 2003. ACM. ISBN 1-58113-723-0. doi: http://doi.acm.org/10.1145/956863.956972.

[55] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management (CIKM'03)*, New Orleans, LA, USA, November 2003.

[56] Y.-R. Lin, J. Sun, P. Castro, R. Konuru, H. Sundaram, and A. Kelliher. MetaFac: community discovery via relational hypergraph factorization. In *Proceedings of the 15th ACM SIGKDD Int'l Conf on Knowledge Discovery and Data Mining*, pages 527–536, 2009.

[57] Y. Liu, R. Jin, and A. K. Jain. Boostcluster: boosting clustering by pairwise constraints. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 450–459, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-609-7. doi: 10.1145/1281192.1281242. URL `http://doi.acm.org/10.1145/1281192.1281242`.

[58] B. Long, Z. M. Zhang, and P. S. Yu. Co-clustering by block value decomposition. In *Proceedings of the 11th ACM SIGKDD Int'l Conf on Knowledge Discovery and Data Mining*, pages 635–640. ACM, 2005.

[59] B. Long, Z. M. Zhang, and P. S. Yu. A probabilistic framework for relational clustering. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 470–479, 2007.

[60] B. Long, P. S. Yu, and Z. Zhang. A General Model for Multiple View Unsupervised Learning. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, 2008.

[61] Q. Lu and L. Getoor. Link-based classification. *In International Conference on Machine Learning*, 2003.

[62] P. K. Mallapragada, R. Jin, A. K. Jain, and Y. Liu. Semiboost: Boosting for semi-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(11):2000–2014, Nov. 2009. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.235. URL `http://dx.doi.org/10.1109/TPAMI.2008.235`.

[63] P. Mandayam Comar, P.-N. Tan, and A. K. Jain. Identifying cohesive subgroups and their correspondences in multiple related networks. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, pages 476–483, 2010.

[64] P. Mandayam Comar, P.-N. Tan, and A. K. Jain. A framework for joint community detection across multiple related networks. *Neurocomputing*, 2011.

[65] P. Mandayam Comar, P.-N. Tan, and A. K. Jain. Simultaneous classification and community detection on heterogeneous network data. In *To appear, Data Mining and Knowledge Discovery(DMKD)*, 2012.

[66] H. Masnadi-Shirazi and N. Vasconcelos. Cost-sensitive boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:294–309, 2011. ISSN 0162-8828. doi: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2010.71.

[67] R. McCreadie, C. Macdonald, and I. Ounis. Crowdsourcing Blog Track Top News Judgments at TREC. In M. Lease, V. Carvalho, and E. Yilmaz, editors, *Proceedings of the Workshop on Crowdsourcing for Search and Data Mining (CSDM) at the Fourth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 23–26, Hong Kong, China, February 2011.

[68] M. Meyerhoff. *Introducing sociolinguistics*. Routledge;, 1 edition, 2006.

[69] M. Narayanan, A. Vetta, E. E. Schadt, and J. Zhu. Simultaneous clustering of multiple gene expression and physical interaction datasets. *PLoS Computational Biology*, 6(4), 2010.

[70] J. Neville and D. Jensen. Leveraging relational autocorrelation with latent group models. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'05)*, pages 322–329, Houston, TX, USA, November 2005.

[71] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69, Feb 2004.

[72] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Phys. Rev. E*, 64, 2001.

[73] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, (45):167–256, 2003.

[74] S. Ozawa, A. Roy, and D. Roussinov. A multitask learning model for online pattern recognition. *IEEE Transactions Neural Networks*, 20:430–445, March 2009. ISSN 1045-9227.

[75] L. Page, S. Brin, R. Motwani, and T. Winograd. Pagerank citation ranking: Bringing order to the web. *Technical report, Stanford University*, 1998.

[76] S. J. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345 –1359, 2010.

[77] A. Popescul and L. Ungar. Statistical relational learning for link prediction. In *Proc. of IJCAI Workshop on Learning Statistical Models from Relational Data*, 2003.

[78] S. Purnamrita, C. Deepayan, and M. Andrew. Theoretical justification of popular link prediction heuristics. In *COLT*, 2010.

[79] M. J. R and N. Heinz. New York: John Wiley, 2nd edition, 1999.

[80] M. Rattigan and D. Jensen. The case for anomalous link discovery. *SIGKDD Explorations*, 7(2):41–47, 2005.

[81] C. M. Richard M. C. McCreadie and I. Ounis. Crowdsourcing a News Query Classification Dataset. In M. Lease, V. Carvalho, and E. Yilmaz, editors, *Proceedings of the ACM SIGIR 2010 Workshop on Crowdsourcing for Search Evaluation (CSE 2010)*, pages 31–38, Geneva, Switzerland, July 2010.

[82] G. Salton and M. J. McGill. Introduction to modern information retrieval. In *McGraw-Hill, Auckland*, 1983.

[83] J. Scripps and P.-N. Tan. Clustering in the presence of bridge-nodes. In *Proceedings of the SIAM International Conference on Data Mining (SDM'06)*, Bethesda, MD, USA, April 2006.

[84] J. Scripps, P. N. Tan, and A.-H. Esfahanian. Exploration of link structure and community-based node roles in network analysis. In *Proceedings of the Seveth IEEE International Conference on Data Mining*, 2007.

[85] J. Scripps, P. Tan, F. Chen, and A.-H. Esfahanian. A matrix alignment approach for link prediction. In *Proceedings of the 19th International Conference on Pattern Recognition*, pages 1–4, 2008.

[86] J. Scripps, P.-N. Tan, and A.-H. Esfahanian. A matrix alignment approach for collective classification. In *Proc of the 2009 Int'l Conf on Advances in Social Networks Analysis and Mining*, Athens, Greece, 2009.

[87] J. Scripps, R. Nussbaum, P. Tan, and A. Esfahanian. Link-based network mining. In M. Dehmer, editor, *Structural Analysis of Complex Networks*. Birkhauser, 2010.

[88] D. Sculley. Combined regression and ranking. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, 2010.

[89] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[90] T. E. Senator. Link mining applications: progress and challenges. *SIGKDD Explorations*, 7(2):76–83, 2005.

[91] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proceedings of CVPR*, 1997.

[92] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[93] M. Soleymani and M. Larson. Crowdsourcing for Affective Annotation of Video: Development of a Viewer-reported Boredom Corpus. In M. Lease, V. Carvalho, and E. Yilmaz, editors, *Proceedings of the ACM SIGIR 2010 Workshop on Crowdsourcing for Search Evaluation (CSE 2010)*, pages 4–8, Geneva, Switzerland, July 2010.

[94] W. Tang, Z. Lu, and I. Dhillon. Clustering with multiple graphs. In *ICDM*, pages 1016–1021, Miami, FL, 2009.

[95] C. Tantipathananandh, T. Y. Berger-Wolf, and D. Kempe. A framework for community identification in dynamic social networks. In *Proceedings of the Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007.

[96] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI02)*, 2002.

[97] B. Taskar, M. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *In Advances in Neural Information Processing Systems 16*, 2003.

[98] B. Taskar, M. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Proceedings of the Neural Information Processing Systems (NIPS 2003)*, 2003.

[99] H. Tong, C. Faloutsos, and J. Y. Pan. Fast random walk with restart and its applications. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 613–622, 2006. URL `http://dx.doi.org/10.1109/ICDM.2006.70`.

[100] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 577–584, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL `http://dl.acm.org/citation.cfm?id=645530.655669`.

[101] P. Wang and C. Domeniconi. Building semantic kernels for text classification using wikipedia. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, 2008.

[102] P. Wang, C. Domeniconi, and J. Hu. Using wikipedia for co-clustering based cross-domain text classification. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 1085–1090, 2008. ISBN 978-0-7695-3502-9.

[103] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)*. Cambridge University Press, 1994.

[104] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[105] Y.-B. Xie, T. Zhou, and B.-H. Wang. Scale-free networks without growth. In *Physica*, volume 387, 2008.

[106] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram. Multi-task learning for classification with dirichlet process priors. *J. Mach. Learn. Res.*, 8:35–63, 2007.

[107] T. Yang, R. Jin, Y. Chi, and S. Zhu. Combining link and content for community detection: a discriminative approach. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 927–936, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-495-9. doi: 10.1145/1557019.1557120. URL `http://doi.acm.org/10.1145/1557019.1557120`.

[108] T. Yang, R. Jin, Y. Chi, and S. Zhu. Combining link and content for community detection: a discriminative approach. In *Proc. of the 15th ACM SIGKDD International Conference on Data Mining*, pages 927–936, Paris, France, 2009.

[109] T. Yang, R. Jin, Y. Chi, and S. Zhu. Combining link and content for community detection: a discriminative approach. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09)*, pages 927–936, 2009.

[110] Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, 18, March 2002.

[111] D. Zhang and D. Shen. Multi-modal multi-task learning for joint prediction of multiple regression and classification variables in alzheimer's disease. *NeuroImage*, 59(2):895 – 907, 2012.

[112] Z. Zhang, T. Li, C. Ding, and X. Zhang. Binary matrix factorization with applications. In *Proceedings of the IEEE Int'l Conf on Data Mining*, pages 391–400, 2007.

[113] L. Zhao, G. Sukthankar, and R. Sukthankar. Robust active learning using crowdsourced annotations for activity recognition. In *AAAI workshop on Human Computation*, 2011.

[114] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, 2004.

[115] D. Zhou, S. Zhu, K. Yu, X. Song, B. L. Tseng, H. Zha, and C. L. Giles. Learning multiple graphs for document recommendations. In *WWW '08*, pages 141–150, 2008.

[116] T. Zhou, L. Lu, and Y.-C. Zhang. Predicting missing links via local information. In *Eur. Phys. J.*, 2009.

[117] S. Zhu, K. Yu, Y. Chi, and Y. Gong. Combining content and link for classification using matrix factorization. In *Proceedings of the 30th Annual Int'l ACM SIGIR Conf on Research and development in information retrieval*, pages 487–494. ACM, 2007.

[118] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. 2002.