SIMILARITY LEARNING IN THE ERA OF BIG DATA

BY

SHIYU CHANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Doctoral Committee:

Professor Thomas S. Huang, Chair
Professor Mark A. Hasegawa-Johnson
Professor Zhi-Pei Liang
Dr. Charu C. Aggarwal, IBM T.J. Watson Research Center

ABSTRACT

This dissertation studies the problem of similarity learning in the era of big data with heavy emphasis on real-world applications in social media. As in the saying "birds of a feather flock together," in similarity learning, we aim to identify the notion of being similar in a data-driven and task-specific way, which is a central problem for maximizing the value of big data. Despite many successes of similarity learning from past decades, social media networks as one of the most typical big data media contain large-**volume**, **various** and high-**velocity** data, which makes conventional learning paradigms and off-the-shelf algorithms insufficient. Thus, we focus on addressing the emerging challenges brought by the inherent "three-Vs" characteristics of big data by answering the following questions: 1) Similarity is characterized by both links and node contents in networks; how to identify the contribution of each network component to seamlessly construct an application orientated similarity function? 2) Social media data are massive and contain much noise; how to efficiently learn the similarity between node pairs in large and noisy environments? 3) Node contents in social media networks are multi-modal; how to effectively measure cross-modal similarity by bridging the so-called "semantic gap"? 4) User wants and needs, and item characteristics, are continuously evolving, which generates data at an unprecedented rate; how to model the nature of temporal dynamics in principle and provide timely decision makings? The goal of this dissertation is to provide solutions to these questions via innovative research and novel methods. We hope this dissertation sheds more light on similarity learning in the big data era and broadens its applications in social media.

To my parents and wife, for their love and support.

ACKNOWLEDGMENTS

I would like to express my appreciation and gratitude to my advisor, Professor Thomas S. Huang, for being such a patient and nurturing advisor and for his constant guidance and support throughout my study at UIUC. I am also grateful to Dr. Charu C. Aggarwal from the IBM T. J. Watson Research Center, who provided me with great opportunities, valuable advice and continuous encouragement. Great appreciation goes to Professor Mark A. Hasegawa-Johnson and Professor Zhi-Pei Liang for serving on my doctoral committee and offering invaluable comments when helping me prepare this dissertation. I also would like to recognize my fellow Image Formation and Processing group members; especially Dr. Guo-Jun Qi and Dr. Zhen Li for their assistance and guidance as well as many insightful suggestions. I am also deeply indebted to my project collaborators Dr. Jiliang Tang and Dr. Jiayu Zhou. It has been my honor to have the opportunity learn from them and have brilliant discussions together. I would like to give special thanks to Yang Zhang, Thomas L. Paine, Pooya Khorrami, Wei Han, Honghui Shi, Ding Liu, Kaizhi Qian and Xinchao Wang for being wonderful colleagues as well as supportive friends. Lastly, I would like to thank my parents, wife and other family members. Without their long-lasting love, support and encouragement, this dissertation would not have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACC	Equal Error Rate Accuracy
AI	Artificial Intelligence
AP	Average Precision
AUC	Area Under Curve
CFSL	Content-based Factorized Similarity Learning
CCA	Canonical Correlation Analysis
CNN	Covolutional Neural Networks
DTL	Distance Transfer Learning
FC	Fully Connected
FSL	Factorized Similarity Learning
HNE	Heterogeneous Network Embedding
IMDB	Internet Movie Database
ItemKNN	Item-based K-Nearest Neighbor
JC	Jaccard's Coefficient
kNN	k-Nearest Neighbor classifier
LAD	Locally-Adaptive Decision
LCMF	Link-Content Matrix Factorization
LUFS	Linked Unsupervised Feature Selection
mAP	Mean Average Precision
NMF	Nonnegative Matrix Factorization

NMI	Normalized Mutual Information
NTF	Nonnegative Tensor Factorization
OCCF	One-Class Collaborative Filtering
PCA	Principal Component Analysis
PMF	Probabilistic Matrix Factorization
PSD	Positive Semi-Definite
PU	Positive-Unlabeled
PUMC	Positive-Unlabeled learning for Matrix Completion
ReLu	Rectified Linear Unit
RBM	Restricted Boltzmann Machines
ROC	Receiver Operating Characteristic
SPU	Streaming Positive-Unlabeled Learning
SSMetric	Semi-Supervised Metric learning
SVM	Support Vector Machines
TF-IDF	Term Frequency - Inverse Document Frequency

CHAPTER 1

INTRODUCTION

Similarity learning - answering the question whether two given objects are similar, or whether object A is more similar to B compared to C - is an important problem with many applications. Frankly speaking, it is a long standing research problem that has been studied since the beginning stage of artificial intelligence (AI) and pattern recognition. In classification, the first algorithm that we learned - k-Nearest Neighbor classifier [1] - uses distance metric as a dissimilarity function to identify the nearest neighbors. Fundamental clustering algorithms, such as the prominent k-means [2], rely on distance measurements between data instances. In information retrieval, candidate results are often ranked according to their relevance to a given query. Clearly, the performance of these methods depends on the quality of the underlying similarity function. As in the saying "birds of a feather flock together," in similarity learning, we hope to identify the pairs of instances that are indeed semantically close, through a data-driven and task-specific way [3].

Although there have been many successes in learning similarity in past decades, as we enter the era of big data, similarity learning faces many new challenges. The first impression when people talk about big data is its size, which is usually referred to in the literature as the "volume". With the booming of social networks, e-commerce, and smart devices in the past ten years, data has increased greatly. According to the International Data Corporation, the number of overall created data had reached 1.8 ZB (approximately equals 10^{21} B) worldwide in 2011, an increase of nine times within five years [4]. Especially, big data related to the services of social media grow rapidly. For instance, Facebook generates log data of over 10 petabyte (PB) per month, while Taobao, a subsidiary of Alibaba, receives data by the tens of terabyte (TB) per day from its online trading [5].

While the amount of data is drastically rising, the variety of data also

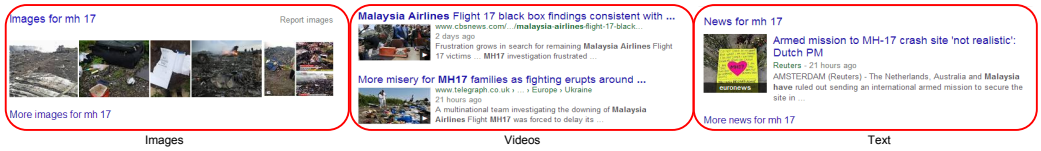


Figure 1.1: Illustration of the heterogeneity of different data sources described by the same topic "Malaysia Airlines MH 17".

brings about many challenges, which demands prompt solutions. "Variety" indicates the various types of data. Social media contains semi-structured and unstructured [6] data in various types such as image, audio, video, webpage and text. For instance, different model types are reported when an event takes place. As a Google search example of "Malaysia Airlines MH 17" illustrates in figure 1.1, relevant results include not only text documents but also images and videos.

Furthermore, "velocity" is another inherent characteristic for big data. In almost all online social media systems, data are generated at an unprecedented rate. For example, Facebook users exceeded one billion in 2012, which doubled from the year before[1]; Netflix gained more than three million subscribers from mid-March 2013 to April 2013[2]; and more than 10 million transactions are made per day in eBay[3]. Such data have distinct properties such as being temporally ordered, continuous and at high-velocity. Data collection and analytics must be rapidly and timely performed to catch the instantaneous need of users and maximize the commercial value of big data.

Often, volume, variety and velocity are refereed as the "three-Vs" for big data [7]. While the problem of similarity learning retains its important role in many modern applications, the game of learning has changed significantly compared to a decade ago. Specifically, many conventional setting and off-the-shelf algorithms are no longer suitable for big data. Thus, in this dissertation, we focus on addressing the emerging "three-V" issues of similarity learning in big data. Due to broad applications of big data, we specifically restrict our regime to the study of social media data, which is considered one of the most representative big data media that are easily accessible. We will show that by combining rigorous mathematics with judicious design, we can make broad impacts on society and uncover natural and social phenomena in the era of big data by similarity learning.

[1] `http://www.businessinsider.com/facebook-350-million-photos-each-day-2013-9`
[2] `https://en.wikipedia.org/wiki/Netflix`
[3] `http://www.webretailer.com/articles/ebay-statistics.asp`

1.1 Big Data Applications

Similarity learning can potentially be beneficial for tasks in which the notion of similarity among data plays an important role. Recently, it has been widely applied to many fields such as computer vision [8, 9], data mining [10, 11, 12, 13, 14, 15], machine learning [16, 17, 18], speech [19, 20], *etc.* In the following, we list several important applications in social media analysis, where similarity learning has been shown to be extremely useful.

Link prediction: The task of link prediction aims to predict missing links in a given network and new or dissolution links in future time, which is important for mining and analyzing the evolution of networks [21]. Link prediction techniques have been widely used in many online social network sites. Facebook and Linkedin recommend missing edges (friendship connections) in their social graph. To solve the link prediction problem, it needs to determine the formation or dissolution probability between all node pairs. A natural and effective way to model such probabilities is to calculate the node-based similarity or relative ranks through either a predefined metric or a data-driven approach. A comprehensive survey on link prediction has been published by Wang *et al.* [22], which includes many state-of-the-art link prediction algorithms via similarity learning.

Community detection: Community detection is often viewed as a clustering task in networks. Communities (also called clusters or modules in the literature) are groups of vertices which probably share common properties and/or play similar roles within the graph. It has been shown that real networks are not random graphs (*i.e.* the probability of having an edge between any pair of nodes is equal). Instead, the probability of connection among node pairs within a same community is much higher than the inter-community connection rate [6, 23]. Identifying graph communities is a popular topic in big data with concrete demands. For instance, clustering large networks can create optimal data structures for efficient graph storage, query and search [24]. Other important real-world applications and emerging techniques are presented in the survey [25].

Recommendation: In recent years, recommendation has become one of the most active research areas driven by enormous industrial needs. The business model for many major technology companies such as Amazon, Netflix and

Pandora heavily relies on their recommender systems. The objective of many recommender systems is to provide the user with the most relevant items according to his/her historical data. Based on the type of user generated data, the task can be further divided into two groups: recommendation with explicit feedback [26] and recommendation with implicit feedback [14, 21]. The ranking of all candidate items is often achieved by measuring the similarity between the user and an item. Applications of similarity learning for recommendation include the work in [26, 27, 28, 29], and interested readers may refer to the surveys [30, 31] for more details.

Information retrieval: Google search engine is the most successful transformation from information retrieval research to our daily life needs. Similar to recommendation, information retrieval algorithms rank candidate documents, images or videos based on their similarity scores to a specific query. Examples include the work of [32, 33].

1.2 Research Challenges

The problem of similarity learning for large-scale, noisy, heterogeneous, and high-velocity social media data is fundamentally different from the conventional settings that have been used over past decades, which requires emerging research efforts. In this section, we list several major challenges posed by the application of similarity learning in the new settings.

Networked data: The availability of vectorized data representations are frequently assumed in conventional similarity learning. They are easy to handle since each data can be viewed as a point residing in a Euclidean space [34, 35]. Thus, the similarity between different data points can be directly measured by an appropriate metric to solve tasks such as classification, clustering and retrieval. Unfortunately, many data sources (*e.g.* Facebook, YouTube, Twitter, *etc.*) in the era of big data cannot be naturally represented as vectorized inputs. Instead, networks or graphs are commonly used to represent these user interactions and online social activities.

Similarity learning in the network environment differs from traditional setups, mainly due to the notion that similarity is characterized by both the link and content. The notion of similarity is reflected by the network connectivity,

content correlation or a combination of the two. Due to social connectivity structures, we note that the nature of data connection plays a vital role in discovering similar node pairs. As a concrete example from sociology, the concept of homophily (*i.e.*, "love of the same") [36] describes the tendency of individuals to associate and bond with similar others, which implies that connected components in networks tend to share common characteristics. On the other hand, metadata can be easily obtained in social networks, which often can be adopted as indicators to identify similar user groups.

Next, we describe a toy example from a real-world scientific author recommendation and retrieval scenario. We show why the direct adoption of content or link metrics only fails to provide good predictions. We consider the top six similar authors calculated from two different aspects of the author *Thomas S. Huang* in the *DBLP-Four-Areas* data set [37], which will be formally introduced in section 3.6.1. Table 1.1 illustrates search results by directly utilizing link weights and content features, respectively. We observed that recommended authors using link information are only Thomas Huang's close collaborators, students, or postdoctoral associates. However, link weights fail to maintain high precision for a long ranking list because of sparsity issues. The main problem is the selection of the proper choice of *indirectly* connected candidates. On the other hand, among authors retrieved from the content source, most of them shared mutual interests for specific scientific topics. One of the drawbacks for such approaches is that each author is usually interested in several research topics or belongs to multiple latent categories. Therefore, the use of a global content measure overlooks the "similarity" in a fine-grid level. From this example, we see that the retrieved results vary a lot from the different perspective of similarity measures. Utilizing either of the two alone is insufficient to retrieve nodes with similar attributes in networks. The scientific goal of similarity learning in networks is to determine the contribution of network components in a task specific fashion for network oriented applications.

Large volume data: The big data era brings huge challenges of data acquisition, storage, management and analytics. Specifically, data are massive and generated at high speed. Therefore, similarity learning needs to be implemented very efficiently and be able to work with one pass of the data [21]. Furthermore, the input streams consist of not only new relations, but

Table 1.1: A motivating example to illustrate the variations in the similarities between nodes from different perspectives.

	Link	Content
Rank 1	Shuicheng Yan	Anni R. Bruss
Rank 2	Brendan J. Frey	Qiang Yang
Rank 3	Xiaoou Tang	Takeo Kanade
Rank 4	Ying Wu	Jaime G. Carbonell
Rank 5	Huan Wang	Rong Jin
Rank 6	Antonio Colmenarez	Raghu Ramakrishnan

also new registered entities which can be introduced to the system for any given time. Therefore, it requires the algorithm to provide responses to any unseen candidates without assuming a fixed number of nodes in the network (out-of-the-sample issue [38]).

Noisy data: Social media data contains much noise. For example, researchers noticed that spammers generate more data than legitimate users [39] due to the autonomous nature of online activities. One the other hand, since users with different backgrounds and knowledge post content freely, only a small amount of data is valuable for task specific learning. The noisy nature of the underlying network poses a great challenge to data acquisition and effective learning. From the similarity learning aspect, many links are not semantically meaningful, especially in online social networks such as Facebook. This is because the cost of connection in Facebook is very low. In this context, it is essential to make the network similarity learning algorithms capable of distinguishing the meaningful aspect of the network characteristics under noisy scenarios.

Varied data: Multimedia data (*e.g.* image, audio, text and video) have been growing rapidly, which is specifically useful in extracting knowledge and understanding user intension. However, multimedia data is heterogeneous, which means that data from different modalities are not directly comparable. Learning similarity between them is confronted with the huge challenge of the semantic differences. For example, two images with similar visual appearances (low-level similarity, *e.g.* color, shape, *etc.*) might not correspond to similar high-level concepts (*e.g.* label information). Such a phenomenon is often called the "semantic gap" [12]. How to bridge the gap is a key for

effectively learning with multimedia inputs.

High-velocity data: The high-velocity characteristic of big data requires learning algorithms to update and respond timely in order to catch users' instant intentions and demands. However, many conventional schemes analyze the new coming data and update their models at regular time intervals (*e.g.* every day or number of days). In addition, for many big data applications, the underlying user preferences (or item characteristics) continuously evolve over time, which can have significant effects on predictions. For instance, in the context of co-authorship prediction, one may change the research interest from one field to another, which decreases the likelihood of connecting to people from the original field. Thus, the learning similarity should capture such signals and timely adapt its prediction accordingly.

1.3 Organization

Different applications in big data era might encounter one or multiple aforementioned challenges. In this dissertation, each chapter specifically targets one or several aforementioned research challenges in an application oriented way. We acknowledge that this dissertation only coversa few aspects of big data analytics. We hope to shed more light on the problem of similarity learning in big data, gain the attention of the research community, and open up new directions in which we can contribute more in the future.

The remainder of this dissertation is organized as follows. In Chapter 2, we introduce some basic and necessary concepts in learning similarity in different types of networks. In Chapter 3, we first introduce a static network representation followed by a detailed investigation of learning similarity in large and noisy networks. In Chapter 4, we turn our view to heterogeneous networks. Specifically, we propose a general framework to convert heterogeneous networks to vectors so that off-the-shelf similarity learning algorithms can be applied to many big data applications. Chapter 5 focuses on learning similarity in high-velocity environments. We unify many big data tasks such as link prediction, search and recommendation to a single steaming Positive-Unlabeled (PU) scheme. Finally, we conclude the dissertation and point out broader impacts and promising research directions in Chapter 6.

CHAPTER 2

BACKGROUND AND PRELIMINARIES

Before proceeding to study the problem of similarity learning in big data environments, we first need to review several basic concepts and well-known algorithms for the purpose of similarity learning in different networks.

2.1 Static Networks

Here, we briefly review existing approaches for learning similarity functions in static networks. In general, similarity learning can be done by either using content, network topology or a combination of these two.

2.1.1 Content-based Similarity Learning

In recent years, there is some emerging research interest in learning content-based similarity in a low-dimensional space such that the regular Euclidean metric is more meaningful in terms of reflecting semantic "closeness" [10]. The first category is supervised metric learning, which is learning a distance metric from the training data with explicit class labels. The representative techniques include the Neighborhood Component Analysis [16] and the Large Margin Nearest Neighbor [17]. However, the performance of the supervised approaches relies heavily on the number of labeled training data examples. This is a problem, because such labels are usually not available in significantly large numbers. Xing *et al.* [18] proposed to use side information instead of class labels. The side information is presented as pairwise constraints associated with input data, which provides weaker information than the exact class labels. In particular, each constraint indicates whether a pair of samples are similar or not. Subsequently, there were several promising research directions, such as Relevance Component Analysis [40] and Infor-

mation Theoretic Metric Learning [41].

However, most of the existing metric learning algorithms do not scale well across various high dimensional learning paradigms. The reason is the size of the distance matrix scales with the square of the dimensionality. Sparse Distance Metric Learning [42] works under pairwise relevance constraints to produce sparse metrics which significantly reduce the number of parameters, so that the time required for learning reduces dramatically. Another issue that makes metric-based similarity learning inefficient for real-world applications is the positive semi-definite (PSD) constraints imposed on the distance matrix. In general, it requires nontrivial PSD programming [43] techniques, and the computational complexity is cubic in the dimensionality of the input data. A recent work proposed by Zhen *et al.*, which is referred to as Locally-Adaptive Decision Learning (LAD) [9], learns a non-isotropic similarity function by a joint model of a distance metric and a locally adaptive thresholding rule. The LAD algorithm relaxes the PSD constraint so that the learned similarity can be negative, if only the relative order is appreciated.

All aforementioned methods assume the availability of vectorized inputs, which often cannot directly apply to these applications with networks involved.

2.1.2 Link-based Similarity Learning

In contrast to content-based similarity learning, link-based methods emphasize network topological structures. The most popular link-based similarity learning method or ranking system is known as the PageRank [32], which is used by the Google search engine. The original Brin and Page model for PageRank uses the hyperlink structure of the web to build a Markov process with a primitive transition probability. A lot of link-based similarity learning approaches are motivated by PageRank including SimFusion [44], Pagesim [45] and the Relational Like-base Ranking [46].

An interesting method, known as SimRank [47], is an iterative PageRank like structure similarity measure in networks. However, SimRank only utilizes the in-link relationships for proximity computation while neglecting the information conveyed from out-links. Zhao *et al.* proposed a P-Rank [48] algorithm which extends SimRank by considering both in-link and out-link

simultaneously. It is worth mentioning that most existing link-based methods rely heavily on the homophily assumption [36], and are insufficient for fully capturing the underlying semantics.

2.1.3 Fusion-based Similarity Learning

Fusion-based similarity learning aims to incorporate both content and link similarity in a seamless way. This issue is often addressed in graph-based semi-supervised learning and recommender systems.

Semi-supervised learning [12] considers the situations where only a few labeled data are available and the vast majority of data remain unlabeled, which has significant impact on pervasive applications in machine learning and data mining. Among many semi-supervised methods, graph-based algorithms are substantially relevant to fusion-based learning, and define a graph where the nodes (feature vectors) are both labeled and unlabeled, and edges reflect connectivity between data samples. Then, similarity can propagate through the graph in a discriminative and transductive way by satisfying the following two requirements: 1) the learned similarity function should satisfy given labeled constraints as much as possible; and 2) similarity function should be smooth across the whole graph. Some representative works include: MinCut [49], Gaussian Random Fields and Harmonic Functions [50], Local and Global Consistency [51], Manifold Regularization [52], Graph Kernels and Laplacian Graph [53], *etc.* The survey in [54] contains comprehensive studies on semi-supervised learning in graphs.

One the other hand, in the task recommender systems, both content and link information are often assumed available. The problem of recommendation can be viewed as learning similarity on a bipartite graph, which one side is the user nodes and the other side is the item nodes. Links between them are the user-item iterations. Often, users/items in recommender systems contain auxiliary information, which describes their intrinsic properties (*e.g.* age, gender, job for users and content descriptions as genre, year of produce for items) by vectors. Research [14] has shown that integrating content-based knowledge enhances the performance on recommendation. In [29, 55], a hybrid movie recommender system is proposed that makes use of both content information acquired from the Internet Movie Database (IMDB) as well as

the collaborative information. A naive Bayes classifier is built to impute the missing entries first, followed by traditional collaborative filtering on the dense pseudo rating matrix. In [56], the authors propose a novel knowledge transfer based algorithm to learn a regression model from the content features to the item latent representations. This approach utilizes existing factorization based techniques [57, 58] to obtain the latent representations of each item with user ratings. A regression model bridges the high-dimensional content features to the low-dimensional latent space. Content-boosted Matrix Factorization [59] incorporates content information directly into the matrix factorization, which assumes the latent item profiles can be further decomposed by a projection of the content vectors. The survey [30] we mentioned above also contains detailed discussion on fusion-based methods.

2.2 Heterogeneous Networks

Recently, many researchers have started to consider similarity measure on heterogeneous networks [60]. Wang *et al.* [61] proposed a model that focuses on analyzing the context of heterogeneous networks. However, their method overlooks the similarity from the network structure perspective. Based on a Markov chain model, Fouss *et al.* [62] designed a distance function termed Euclidean Commute Time Distance. Unfortunately, the absence of a path constraint makes this method cannot capture the subtle semantics in heterogeneous networks. Sun *et al.* proposed PathSim [63] by considering semantics in meta paths constituted by different-typed objects. The drawback for PathSim is that it only considers the similarity of same-typed objects based on symmetric paths, which ignores many valuable asymmetric paths. In order to evaluate the relevance of different-typed objects, Shi *et al.* [64] proposed HeteSim to measure the relevance of any object pairs under arbitrary path, which makes it is suitable for a wide range of applications.

Another branch of calculating similarity in heterogeneous networks utilizes network embeddings [15]. These models often transfer the problem as learning an embedding of the entities, which algebraically corresponds to a matrix factorization problem of observed relationships. Zhu *et al.* [65] proposed a joint factorization approach on both link and document-term frequency matrix for Web page categorization. Similar concepts also include [66, 67].

However, these models focus only on single relations that do not adapt to heterogeneous settings. A natural extension of these methods stacks the relational matrices, and then applies conventional tensor factorization [68, 69]. The disadvantage of such multi-relational embeddings is the inherent sharing of parameters between different terms, which does not scale to large graphs. A nonlinear embedding model is proposed by Yuan *et al.* [70], which uses Restricted Boltzmann Machines (RBM) for cross-model link analysis. However, it requires a feature vectorization step, which results in information loss. A survey of heterogeneous information network analysis can be found in [60].

2.3 Streaming Networks

In the era of big data, the inputs for many online systems arrive in a streaming fashion [71, 72]. These data flow into a system in vast volumes, change dynamically and are possibly infinite [6, 73]. When the data are of large volume, they cannot be stored in traditional database systems. Moreover, most systems may only be able to access the stream once. This poses great computational and mining challenges. There have been many works on efficient methods for mining data streams, which specifically work with one pass of the data. Many research problems that are closely related to similarity learning in streaming graphss have been explored, which include counting triangles [74], finding common node-neighbors [75], estimating PageRank scores [76], clustering graph streams [77], outlier detection [78], mining dense structural patterns [79], *etc.* Meanwhile, the stream mining process is dynamic since user behaviors as well as item characteristics may evolve over time. Many stream mining algorithms focus on the evolution of the underlying data [26, 21]. Readers could refer to these books [80, 6] and surveys [81, 82, 83] for details.

2.4 Related Topics

We mention here three general machine learning topics that are related to our research, which served as important foundations for our proposed methods.

2.4.1 Matrix Factorization

Matrix factorization is one of the most popular methods in matrix completion and recommendation. Typically, the factorization assumes that there are low rank distributions in space, and a low rank approximation is utilized to regularize the factorization process. The fundamental problem is to fill out the missing entries of the utility matrix with sparse observations. Traditional approaches include Low-rank Matrix Fitting [84], Nonnegative Matrix Factorization (NMF) [85] and Probabilistic Matrix Factorization (PMF) [57], which fit a probabilistic distribution for the matrix. In the domain of collaborative filtering, which learns the similarities between different entries, the social hints are also considered in addition to link structures [86, 87]. These approaches are referred to as the social matrix factorization. Other approaches try to incorporate content similarities into the factorization, and a typical extension is the Collaborative Topic Modes [88].

2.4.2 Deep Learning

Deep learning plays a central role in an important shift in machine learning research that emphasizes feature learning from raw data. It has become increasingly important in speech recognition, object recognition/detection, and natural language processing. Recent advances in deep learning have benefited from a confluence of factors, such as the availability of large-scale datasets, computational resources, and advances in both unsupervised and supervised training algorithms. Unsupervised deep learning, often referred to as "pre-training" [89], provides robust initialization and regularization with the help of unlabeled data, which is copiously available. For example, Hinton and Salakhutdinov [90] first employed layer-wise initialization of deep neural networks with the use of RBMs. A similar approach is weight initialized with auto-encoders found by Bengio *et al.* [91]. More recently, supervised learning with multilayered neural networks has been proven possible. Covolutional Neural Networks (CNN) [92], with the method of Dropout [93] and Rectified Linear Unit (ReLu) [94], has shown particular promise. The combination of these recent advances keeps breaking the record of the ImageNet challenges [92, 95]. The development of deep learning is too fast to be comprehensively introduced here; we recommend that interested readers see the survey [96]

written by LeCun, Bengio and Hinton, and pay close attention to arXiv[1].

2.4.3 PU Learning

The problem of PU learning is first studied in binary classification, where training examples only consist of positive labels. Two general approaches have been previously proposed to handle such "one-side" measurements. The first approach involves iterating between two steps, which are 1) identifying possible negative samples (some approaches also include positive ones [97]) from unlabeled data, and 2) applying standard binary classification methods on negative samples identified in the previous step [98]. The other approach assigns weights to each unlabeled datum, and then trains a classifier with the unlabeled data interpreted as weighted negative samples [99, 100]. Although many algorithms have been well developed for classification with PU inputs, they assume data are in the form of vectorized representations, which is not applicable to problems where only network topology information is available.

Matrix factorization [57, 84, 85, 101, 102] is one of the most popular approaches to link prediction or recommendation since it does not require any auxiliary content features. However, most of the existing approaches are not specifically designed for the PU inputs. An important variant of the matrix completion problem is to recover an underlying matrix from one-bit quantizations, which is an instance of PU learning. Davenport *et al.* [103] first analyze one-bit matrix completion under a uniform sampling model, where observed entries are assumed to be sampled uniformly at random. However, in data mining applications such as collaborative filtering, the uniform sampling model is overidealized. An improved method has been proposed in [104], which replaces the uniform sampling assumption with the max-norm as a convex relaxation for the rank. Recently, Hsieh *et al.* [105] proposed a method termed PU Learning for Matrix Completion (PUMC), which contains well developed theories on performing one-bit matrix completion by assigning different costs to observed and unobserved entries in the objective. Similar ideas [106, 107] have also been used in recommender systems, albeit heuristically. It is worth mentioning that all these methods are batch models without considering any temporal aspect of the data.

[1] `https://arxiv.org/`

CHAPTER 3

LARGE VOLUME DATA

Nowadays, networks are ubiquitous in the context of data mining and information retrieval applications. Social and technical information systems usually exhibit a wide range of interesting properties and patterns such as interacting physical, conceptual and societal entities. Each individual entity interchanges and influences others in the context of this interconnected network. Similarity learning as a central tenant of network mining research in the era of big data will be first investigated, for large and noisy networks, in this chapter.

3.1 Static Network Representation

The two fundamental components defining a network topology are nodes and edges. In this chapter, we model any given network as a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents a set of nodes/vertices and $\mathcal{E}$ represents the edges between these nodes. We denote the vertices by $\mathcal{V} = \{v_1, \ldots, v_n\}$ and edges by $\mathcal{E} = \{e_1, \ldots, e_m\}$. Thus, there are a total of n nodes and m directed edges. The directed assumption is without loss of generality, because undirected networks can be easily converted into a directed framework, by simply replacing undirected links by two directed edges. We further assume, that two additional types of information are available. One of them corresponds to link weights and the other one corresponds to content features. The weight of a link indicates the strength of the connection, while the content uniquely describes node characteristics. Let $\mathcal{L} = \{l_1, \ldots, l_m\}$ represent the link weights associated with the corresponding edges $\{e_i\}$ in the network, where each $l_i \in \mathbb{R}$, $\forall i = \{1, \ldots, m\}$. Similarly, let $\mathcal{C} = \{c_1, \ldots, c_n\}$ be the set of content features represented by a vector in some vector space in $\mathbb{R}^d$, so that every $v_i \in \mathcal{V}$ is associated with a $d-$dimensional content vector de-

noted by c_i. In summary, we characterize a network using the representation $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{C}, \mathcal{L})$, which includes the graph structure, content and link features.

3.2 Weakly Supervised Similarity Learning in Networks

As the example in table 1.1 shows, either content-based or link-based features alone are insufficient to retrieve nodes with similar attributes in networks. Therefore, we seek a unified learning framework that considers both link and content information. However, the notion of similarity is subjective and task-specific, and correctly understanding it is critical to decision making.

We observe that, in many social applications, although obtaining absolute "label" of each node is hard, the relative ordered information (intentional knowledge) can be easily generated. Such information is often termed as weakly supervised information [108], which could be generated from the user clicking behaviors on the web or transferred from other domains of sources. Mathematically, the weak supervision is modeled by triplet constraints of the form:

$$\mathcal{S} = \{(v_i, v_j, v_k) : (v_i \text{ and } v_j) \text{ more similar to } (v_i \text{ and } v_k)\}.$$

Such information is extremely useful for understanding users' intentions in order to identify the task-specific notion of being similar in a network. Thus, we reveal the problem of similarity learning by also integrating the limited supervision. To achieve this goal, next we will introduce a novel factorization based scheme. Our approach models the similarity learning as a matrix completion problem [57, 84, 85], where it aims at supervised learning the correlation between different nodes using both link and content information so that the completed similarity matrix will correctly reflect the homogeneity between different nodes.

3.2.1 Parameterizations and Constraints

In order to model the similarity learning as a matrix completion problem, we formulate $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{C}, \mathcal{L})$ in matrix form. Let $C \in \mathbb{R}^{n \times d}$ and $L \in \mathbb{R}^{n \times n}$

represent the content and link matrices, which are defined as follows. Each row $C_{i\cdot}$ of the content matrix C is the corresponding feature vector $c_i \in \mathcal{C}$. If the link weight $l_p \in \mathcal{L}$ associates with edge $e_p \in \mathcal{E}$ which connects nodes v_i and $v_j \in \mathcal{V}$, then the L_{ij} entry in the link matrix L will be l_p. A nonzero entry L_{ij} in L indicates that a link exists from the node v_i to v_j, with a weight equal to the strength of the link. It is worth pointing out that both C and L are typically very sparse in practice.

The target is to learn a matrix $S \in \mathbb{R}^{n\times n}$, which reflects the information in both L and C. The (i, j)th entry of S measures the similarity from node v_i to v_j. The similarity matrix S is not necessarily symmetric, because similarity is usually non-isotropic across the network. Thus, we do not explicitly constrain the symmetry of S, in order to make our model more general.

On the other hand, the triplet supervision is modeled as constraints for the space of S, *i.e.*, the similarity matrix S has to obey the supervision as much as possible. If the supervision suggests that nodes v_i and v_j are more similar to each other than nodes v_i and v_k, the learned similarity has to reflect that fact by enforcing $S_{ij} > S_{ik}$. However, in terms of mathematical abstraction, the strict order relationship is not a compact set regularizing the space of S. Almost all existing optimization approaches do not favor the open set constraints. We leverage the problem by each constraint as a closed half-space. Specifically, we require that S has to be in the set $\mathcal{T}$, which is defined as follows:

$$\mathcal{T} \doteq \{S : S_{ij} \geq S_{ik} + c,\ \forall (v_i, v_j, v_k) \in \mathcal{S}\}. \tag{3.1}$$

Here, c is the margin controlling the minimal separability of the similar entries. The value of c can be chosen arbitrarily, since the order between candidate nodes is more important than the actual similarity value at each entry of S. Throughout this chapter, we set c to be equal to 1 for simplicity. Moreover, the following convexity result holds, and the proof can be found in appendix A.1.

Lemma 3.2.1 *The set $\mathcal{T}$, as defined in equation (3.1), is convex.*

3.2.2 Integrate Link and Content

As is generally the case for matrix completion problems, we assume that the rank of S is much less than the number of nodes n in the given network. This is a very natural assumption because the number of latent factors characterizing different nodes is much smaller than the number of nodes. However, unlike existing matrix completion problems, S also satisfies some partial order constraints. The minimum number of latent topics, which allows S to satisfy all the constraints, indicates the intrinsic rank of the similarity matrix. Both content and link data encoded in the network are traded as side information, to enhance the factorization, followed by supervised knowledge.

To utilize all available information, let S be a completed matrix using both content information C and link weight matrix L. We factorize S as $S \cong UV$, where $U \in \mathbb{R}^{n\times r}$ and $V \in \mathbb{R}^{r\times n}$ are two low-rank matrices such that $r \ll n$. $\|S-UV\|_F^2$ penalizes the error by approximating S as the product of two low-rank factors U and V, where $\|\cdot\|_F$ is the Frobenius norm of a given matrix, where $\|X\|_F = \sqrt{tr(XX^T)}$ and $tr(\cdot)$ represents the trace of the matrix.

The link information contributes to similarity learning through the following term:

$$\|\mathcal{P}_\Omega(S) - \mathcal{P}_\Omega(L)\|_F^2, \tag{3.2}$$

where Ω is the index set for the observed elements and the projection $\mathcal{P}_\Omega$ is an orthogonal projector defined in [109]: the (i,j)th element of $\mathcal{P}_\Omega(L)$ is equal to L_{ij} if $(i,j) \in \Omega$ and zero otherwise. In other words, we propagate the link information through its non-zero feature weights. This is done so that the model will have consistent values as suggested by the link features. This term ensures that the similarity matrix S is influenced by the local topological structure.

Furthermore, to encode the content information in our model, we assume that the content matrix C can be factorized as two low-rank matrices, that is a shared U and a basis matrix W, where $W \in \mathbb{R}^{r\times d}$. The following equation ensures the propagation of similarity information from C to S.

$$\|S - UV\|_F^2 + \|C - UW\|_F^2. \tag{3.3}$$

Note that S has already encoded the link information through equation (3.2). The intuition behind these two terms in equation (3.3) is that the projec-

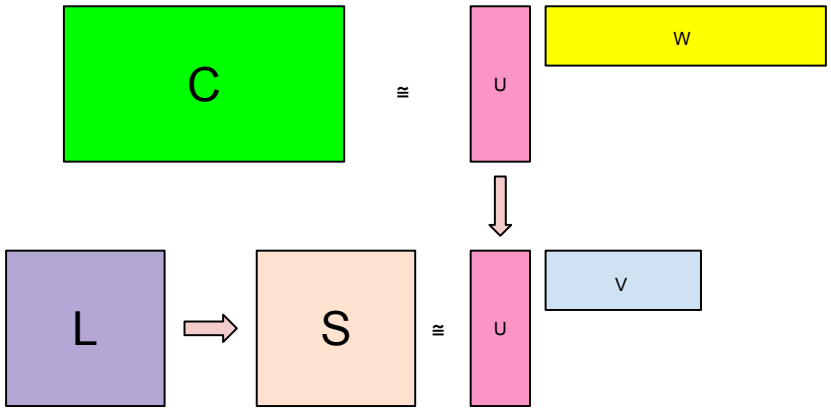


Figure 3.1: An idea illustration for integrating different information sources in networks.

tions from link and content to a common latent space are identical. If we assume that both V and W are orthonormal, then we multiply V^T and W^T on both sides of the equations $S = UV$ and $C = UW$. We obtain the following: $SV^T = U$ and $CW^T = U$. The similarity matrix S, which encodes the link information and the content matrix C, is projected into a common subspace U through projections V^T and W^T. Therefore, the content and link information can be bridged coherently using the aforementioned scheme, so that the learned similarity matrix S is consistent with both content and link information globally and locally. A graphical illustration of how different information sources are fused and transferred to contribute to learning node-based similarity is shown in figure 3.1.

We now integrate all the aforementioned parts into a coherent learning framework as:

$$\begin{aligned} &\min_{U,V,W,S} \ \|\mathcal{P}_\Omega(S) - \mathcal{P}_\Omega(L)\|_F^2 + \lambda_1 \|S - UV\|_F^2 + \lambda_2 \|C - UW\|_F^2 \\ &\text{subject to: } S \in \mathcal{T}, VV^T = I_r, WW^T = I_r. \end{aligned} \tag{3.4}$$

However, the objective in equation (3.4) has two problems, which lead to inefficient optimization algorithms. The first problem is that the first term in the above objective function contains a projection of non-zero entries in the link matrix. $P_\Omega(L)$ can be viewed as indicator function of all non-zero entries of L, which is discrete. Integer programming solvers are usually quite slow. To alleviate these challenges, we introduce a transition variable $T \in \mathbb{R}^{n \times n}$

acting as a bridge to transfer knowledge from L to S. Then, we are able to convert the projection / indicator term in equation (3.4) to a new set of constraints on T. Another issue is the orthonormal constraints on both V and W. These constraints not only introduce more non-convexity into the objective, but also make the algorithms more complex.

Alternatively, we can relax the orthogonal constraint. To prevent overfitting, we introduce Frobenius norms on both V and W. To this end, we reformulate objective function (3.4) as follows:

$$\begin{aligned} \min_{U,V,W,T,S} \; & \|S-T\|_F^2 + \lambda_1\|S-UV\|_F^2 + \lambda_2\|C-UW\|_F^2 \\ & + \lambda_3(\|V\|_F^2 + \|W\|_F^2) \\ \text{subject to: } & \mathcal{P}_\Omega(L) = \mathcal{P}_\Omega(T), S \in \mathcal{T}. \end{aligned} \tag{3.5}$$

3.2.3 Optimization

In this subsection, we demonstrate that the optimization problem in equation (3.5) can be solved efficiently and effectively using the block coordinate descent method [43], which seeks the optimal value for one particular variable, while fixing others. Though the formulation is non-convex, each subproblem in block coordinate descent is convex. The key here is in solving for each of the variable sets U, V, W, T and S, while keeping the others fixed.

Solving for U:
Fixing parameters V, W, T, S to optimize U, the objective function (3.5) reduces to a standard convex unconstrained quadratic program as follows:

$$\min_U \; \lambda_1\|S-UV\|_F^2 + \lambda_2\|C-UW\|_F^2. \tag{3.6}$$

By determining the derivative of the aforementioned objective with respect to U, and setting it to zero, we obtain:

$$-2\lambda_1(S-UV)V^T - 2\lambda_2(C-UW)W^T = 0. \tag{3.7}$$

We can obtain an analytic solution for the global minimum:

$$U^* = (\lambda_1 SV^T - \lambda_2 CW^T)(\lambda_1 VV^T + \lambda_2 WW^T)^\dagger, \tag{3.8}$$

where $(\cdot)^\dagger$ indicates the pseudo-inverse for a given matrix.

Solving for V:
Similar to solving for U, the matrix V can be solved as a standard unconstrained ridge regression problem, and the objective function can be written as follows:

$$\min_V \lambda_1 \|S - UV\|_F^2 + \lambda_3 \|V\|_F^2. \tag{3.9}$$

As in the previous case, we can determine the first order derivative of the objective function in equation (3.9) with respect to V to be zero as follows:

$$-2\lambda_1 U^T (S - UV) + 2\lambda_3 V = 0, \tag{3.10}$$

The aforementioned equation can be solved in order to obtain a global minimum for V.

$$V^* = (U^T U + \tfrac{\lambda_3}{\lambda_1} I_r)^{-1} U^T S, \tag{3.11}$$

where I_r is an identity matrix of size $r \times r$.

Solving for W:
Solving for W is almost identical to solving for V. By fixing U, V, T and S, we can write the objective function and the analytical solution for the optimal value of W as follows:

$$\min_W \lambda_2 \|C - UW\|_F^2 + \lambda_3 \|W\|_F^2. \tag{3.12}$$

The optimal value for W is as follows:

$$W^* = (U^T U + \tfrac{\lambda_3}{\lambda_1} I_r)^{-1} U^T C. \tag{3.13}$$

Solving for T:
When we solve for T, while keeping the remaining parameters fixed, we obtain a constrained least squares minimization problem:

$$\min_T \ \|S - T\|_F^2 \quad \text{s.t.:} \quad \mathcal{P}_\Omega(L) = \mathcal{P}_\Omega(T). \tag{3.14}$$

The equality constraints ensure that non-zero entries of the link matrix L are consistent with the corresponding position on T. Since it is a convex problem, the standard technique for solving equation (3.14) first sets $T = S$, and then

applies the orthogonal projection on T. In particular, we set the entries of T in Ω to be the same, as the corresponding value of L. The compressed analytical solution for S can be written as $T^* = S + (\mathcal{P}_\Omega(L) - \mathcal{P}_\Omega(S))$.

Solving for S:
At this point, we can also solve for S, so that equation (3.5) is minimized. To do so, we obtain the following optimization problem:

$$\min_S \ \|S - T\|_F^2 + \lambda_1 \|S - UV\|_F^2 \quad \text{s.t.: } S \in \mathcal{T}. \tag{3.15}$$

The objective function can be further compressed by a least square term as $\|S - \frac{1}{1+\lambda_1}(T + \lambda_1 UV)\|_F^2$. Since the set $\mathcal{T}$ is a convex set, the problem in equation (3.15) is again a convex constrained optimization problem, which can be solved using projected gradient methods [43]. The definition of a projection is given in 3.2.3. Then, the proximal operator associated with equation (3.15) is in the form of projecting a point to the intersection of a set of halfspaces $\mathcal{T} = \cap_{i=1}^{|S|} T_i \neq \emptyset$, which can be solved using proximal splitting methods [110]. Moreover, we observe that our objective is a simple projection problem, and thus we can use the successive projection algorithm to solve it efficiently [111]. This has the effect of avoiding expensive line search procedures. The optimal S is obtained by first setting it as $\frac{1}{1+\lambda_1}(T + \lambda_1 UV)$, then projecting it onto the convex set $\mathcal{T}$. We now provide a closed form solution to the projection into each set T_i in theorem 3.2.2, where the formal proof can be found in appendix A.2.

Definition A mapping $\Pi_\mathcal{T} : \mathbb{R}^{n\times n} \to \mathcal{T}$ is a projection associated with convex set $\mathcal{T}$, if it satisfies for any $S \in \mathbb{R}^{n\times n}$, $\Pi_\mathcal{T}(S)$ is the unique matrix in $\mathcal{T}$ that is closest to S, *i.e.*,

$$\|S - \Pi_\mathcal{T}(S)\| \leq \|S - S'\|, \quad \forall\ S' \in \mathcal{T}, S \in \mathbb{R}^{n\times n}$$

with equality if and only if $S' = \Pi_\mathcal{T}(S)$.

Theorem 3.2.2 *Suppose that $T_m = \{S : S_{ij} \geq S_{ik} + 1\}$. Then, for any $S \in \mathbb{R}^{n\times n}$ the projection from S to the convex set $\mathcal{T}_m$ is as follows:*

$$\Pi_{\mathcal{T}_m}(S) = S^* = S \quad \textit{if} \quad S \in \mathcal{T}_m,$$

Algorithm 1: Factorized Similarity Learning

Input: Content matrix C, link matrix L and ordered constraint set $\mathcal{T}$
Output: Similarity matrix S

```
Initialize: U, V, W, T and S
repeat
    U = (λ1 S V^T − λ2 C W^T)(λ1 V V^T + λ2 W W^T)^†;
    V = (U^T U + (λ3/λ1) I_r)^{-1} U^T S;
    W = (U^T U + (λ3/λ1) I_r)^{-1} U^T C;
    T* = S + (P_Ω(L) − P_Ω(S));
    S = (1/(1+λ1)) (T + λ1 U V);
    Slice S in row-wise into {S_i.}_{i=1}^n to compute parallel;
    for i = 1 ... n do
        repeat
            if S_ij < S_ik + 1 ∀(i, j, k) ∈ S then
                S_ij = (1/2)(1 + S_ij + S_ik)
                S_ik = (1/2)(−1 + S_ij + S_ik)
            end
        until all constraint satisfied;
    end
until converge or maximum iteration exceed;
return S
```

Furthermore, if $S \notin \mathcal{T}_m$, then the following is true:

$$\Pi_{\mathcal{T}_m}(S) = S^* = \begin{cases} S^*_{ij} = \frac{1}{2}(1 + S_{ij} + S_{ik}) \\ S^*_{ik} = \frac{1}{2}(-1 + S_{ij} + S_{ik}) \\ S^*_{pq} = S_{pq} \quad \forall \{p, q\} \neq \{i, j\} \text{ and } \{i, k\}. \end{cases}$$

To conclude this subsection, we illustrate the optimization scheme for the proposed method in algorithm 1 and name it as the Factorized Similarity Learning (FSL).

3.3 Large-scale Network Handling

For a large-scale network, most commodity hardware cannot hold the similarity matrix S in main memory. This situation is typically arrived at when the number of nodes exceeds 30,000. In order to alleviate this issue, we will show that the proposed method can be easily formulated in a divide and

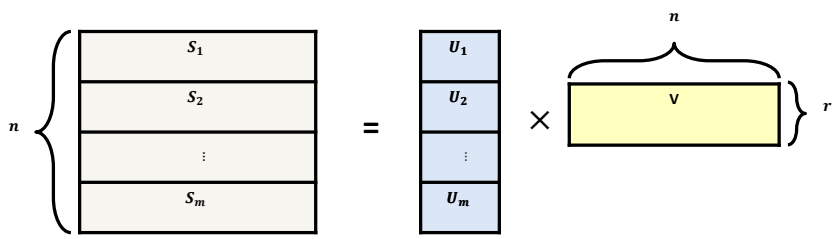


Figure 3.2: Large-scale matrix handling.

conquer framework.

We first slice the similarity matrix S in row-wise fashion, into different sub-matrices $S_1, \ldots S_m$, where each $S_i \in \mathbb{R}^{(n/m)\times n}$. Then, each S_i can be further expressed as $S_i = U_i V$, where each S_i corresponds to a $(n/m) \times r$ matrix U_i. From the block-wise matrix multiplication, we know if we stack each U_i in column-wise fashion, and multiply by V, the result will be exactly equal to the original $n \times n$ similarity matrix S_i. Doing so provides significant memory efficiency gain. Instead of storing an $n \times n$ matrix S, we only require $(n/m) \times n$ floating point space. In an extreme case of $n = m$, we achieve the lowest memory cost. Figure 3.2 provides a visualization of extending the proposed method into a large-scale framework.

The mathematical abstraction can be directly derived from equation (3.5) as follows:

$$\begin{aligned} \min_{U_i,V,W,T_i,S_i,\forall i} \quad & \sum\nolimits_{i=1}^{m} \|S_i - T_i\|_F^2 + \lambda_1 \sum\nolimits_{i=1}^{m} \|S_i - U_i V\|_F^2 \\ & + \lambda_2 \sum_{i=1}^{m} \|C_i - U_i W\|_F^2 + \lambda_3(\|V\|_F^2 + \|W\|_F^2) \qquad (3.16) \\ \text{subject to: } & \mathcal{P}_\Omega(L_i) = \mathcal{P}_\Omega(T_i), S_i \in \mathcal{T}_i \ \ \forall i. \end{aligned}$$

Here, C_i, L_i and T_i are the corresponding sliced content, link and bridging matrices. The overall result is that neither the network information, nor the completed similarity matrix S will be stored in main memory as a whole piece, and the memory can be managed much more efficiently.

Solving for U_i, T_i and S_i:

The process of solving for each U_i, T_i and S_i uses a similar approach. Here, we provide a detailed optimization scheme for U_i and the idea can be easily extended to solve for T_i and S_i.

Calculating U can be seen as optimizing m sub-problems for each U_i (at a smaller scale), which has no interdependency. Moreover, the solution for U_i is exactly the same as before:

$$U_i^* = (\lambda_1 S_i V^T - \lambda_2 C_i W^T)(\lambda_1 V V^T + \lambda_2 W W^T)^\dagger. \tag{3.17}$$

Solving for V and W:
Solving for V is slightly different from the case when we treat matrices S and U as whole. The corresponding Equation (3.9) is transformed as follows:

$$\min_V \lambda_1 \sum\nolimits_{i=1}^m \|S_i - U_i V\|_F^2 + \lambda_3 \|V\|_F^2. \tag{3.18}$$

The optimal analytical solution of V is as follows:

$$V^* = \left(\sum\nolimits_i^m U_i^T U_i + \tfrac{\lambda_3}{\lambda_1} I_r\right)^{-1} \left(\sum\nolimits_i^m U_i^T S_i\right). \tag{3.19}$$

The optimal value of W can be calculated in a similar manner as follows:

$$W^* = \left(\sum\nolimits_i^m U_i^T U_i + \tfrac{\lambda_3}{\lambda_1} I_r\right)^{-1} \left(\sum\nolimits_i^m U_i^T C_i\right). \tag{3.20}$$

MapReduce:
The bottleneck of efficient learning is at the step of updating S or S_i in both conventional and large-scale formulations in equation (3.15) and (3.16) respectively. However, the proposed FSL algorithm is able to decouple the row updates of the similarity matrix S, involving supervised projection. Essentially, this can be easily fit into a MapReduce framework to significantly boost the training efficiency. Moreover, for the large-scale formulation in equation (3.16), the low-rank matrices U_i, bridging matrices T_i and the similarity matrix S_i can also be handled in parallel to reduce the running time. While we present these ideas as possibilities for future exploration, a detailed discussion is beyond the scope of this dissertation. We refer interested readers to [112] and [113] for background on relevant big-data frameworks.

3.4 Noisy Data

Real-world data always contain a significant amount of noise, which could be extremely detrimental to the algorithms. In this section, we explicitly consider the case where the available supervision is noisy. We show how the proposed method can be integrated with noisy intentional knowledge to yield reliable predictions.

We previously modeled the supervised knowledge on different samples as a set of triplet constraints $\mathcal{S}$, in which each element in the constraint set is in the form (v_i, v_j, v_k). Specifically, each triplet supervision provides the similarity information on two pairs of nodes with the same query node. When the noise increases, similarity learning could result in poor quality. We illustrate the problem of noisy supervision with a toy example.

Suppose that four different nodes a, b, c, d are given, and the correct underlying similarity order of using a as a query is that $(a, b) > (a, c) > (a, d)$. If $\{(a, b, c), (a, c, d)\}$ is given as the constraint set $\mathcal{S}$, we can order the candidate nodes b, c, d correctly with respect to reference a. With noisy supervision examples, such as $\{(a, b, c), (a, d, b)\}$ or $\{(a, b, c), (a, d, c), (a, c, d)\}$, the ranking result will either be in an incorrect order, or may have no feasible solution. The inconsistent supervision provides no feasible solution of $S \in \mathcal{T}$ in Equation (3.5).

The aforementioned toy example suggests that the constraints should be relaxed with the use of slack variables ξ_{ijk}. Intuitively, these slack variables can account for the noise in the objective function. Therefore, the modified optimization problem is as follows:

$$\begin{aligned}
\min_{U,V,W,T,S,\xi_{ijk}} \quad & \|S - T\|_F^2 + \lambda_1 \|S - UV\|_F^2 + \lambda_2 \|C - UW\|_F^2 \\
& + \lambda_3 (\|V\|_F^2 + \|W\|_F^2) + \lambda_4 \sum\nolimits_{(i,j,k)\in\mathcal{S}} \xi_{ijk} \\
\text{subject to: } & \mathcal{P}_\Omega(L) = \mathcal{P}_\Omega(T), \xi_{ijk} \geq 0, \\
& S_{ij} - S_{ik} \geq 1 - \xi_{ijk} \quad \forall (i, j, k) \in \mathcal{S}.
\end{aligned} \tag{3.21}$$

It is worth mentioning that the core idea behind such a large-margin relaxation is similar to the formulation of Support Vector Machines (SVM) [114].

A naive way to solve equation (3.21) is to use stochastic sub-gradient

descent [115] by converting the last two constraints as a penalty term in the objective.

$$\begin{aligned}\min_{U,V,W,T,S} \quad & \|S-T\|_F^2 + \lambda_1\|S-UV\|_F^2 \\ & + \lambda_2\|C-UW\|_F^2 + \lambda_3(\|V\|_F^2 + \|W\|_F^2) \\ & + \lambda_4 \sum\nolimits_{(i,j,k)\in\mathcal{S}} \max\{0, 1 - S_{ij} + S_{ik}\} \\ \text{subject to: } & \mathcal{P}_\Omega(L) = \mathcal{P}_\Omega(T).\end{aligned} \tag{3.22}$$

Here, λ_4 regulates the noise penalty. The term associated with λ_4 is the hinge loss [114].

To solve the optimization problem in equation (3.22), we follow a similar procedure, as illustrated in algorithm 1 by the block coordinate descent method. The only difference is that we compute the sub-gradient at the step of solving S instead of using the projected gradient methods. By fixing other parameters to compute the optimal value of S, we obtain:

$$\begin{aligned}\min_{S} \quad & f(S) = \|S-T\|_F^2 + \lambda_1\|S-UV\|_F^2 \\ & + \lambda_4 \sum\nolimits_{(i,j,k)\in\mathcal{S}} \max\{0, 1 - S_{ij} + S_{ik}\}.\end{aligned} \tag{3.23}$$

This is an unconstrained quadratic programming problem. Furthermore, one of the sub-gradients of $f(S)$ is as follows:

$$\begin{aligned}\tfrac{\partial f(S)}{\partial S} &= 2(S-T) + 2\lambda_1(S-UV) \\ &+ \lambda_4 \sum\nolimits_{(i,j,k)\in\mathcal{S}} \mathbb{1}\{1 - S_{ij} + S_{ik} \geq 0\}(E_{ik} - E_{ij}).\end{aligned} \tag{3.24}$$

Here, $\mathbb{1}(\cdot)$ is an indicator function, and $E_{ij} = e_i^T e_j$. Moreover, e_i is the standard unit vector which is a $1 \times n$ vector with only the ith entry set to one, and zero otherwise.

3.5 Speed the Learning up

Comparing the formulation in equation (3.5) to (3.21), similar to SVM, the noisy version always has the advantage in terms of robustness. Thus, in this section, we derive a dual form of the optimization problem (3.23) which possesses an efficient solution to make the algorithm even more suitable for

big data.

Using the slack variables $\{\xi_{ijk}\}$, equation (3.23) is equivalent to

$$\begin{aligned} &\min_{S} \ \|S-T\|_F^2 + \lambda_1\|S-UV\|_F^2 + \lambda_4 \sum\nolimits_{(i,j,k)\in\mathcal{S}} \xi_{ijk}, \\ &\text{subject to: } \xi_{ijk} \geq 0, \quad S_{ij} - S_{ik} \geq 1 - \xi_{ijk} \quad \forall (i,j,k) \in \mathcal{S}. \end{aligned} \tag{3.25}$$

Note that each row of S in the primal problem (3.25) can be optimized separately, since the rows of S are independent of each other in both the objective function and the constraints. Similar to what we discussed in the subsection 3.3, solving S in a row-wise manner significantly facilitates large scale applications and benefits from parallel computing. Let S_i denote the i-th row of S, then the optimization problem for each S_i is written as:

$$\begin{aligned} &\min_{S_i} \ \|S_i-T_i\|_2^2 + \lambda_1\|S_i-U_iV\|_2^2 + \lambda_4 \sum\nolimits_{(i,j,k)\in\mathcal{S}} \xi_{ijk}, \\ &\text{subject to: } \xi_{ijk} \geq 0, \ S_{ij} - S_{ik} \geq 1 - \xi_{ijk}, \end{aligned} \tag{3.26}$$

which is a constrained convex optimization problem. It can be solved by its dual problem due to the strong duality according to Slater's condition [43]. In the sequel we show that the dual problem is a box constrained quadratic programming problem which can be solved efficiently by the coordinate descent algorithm. As opposed to the subgradient method for the primal problem, the limited inequality constraints leads to a dual problem that can be solved much faster by coordinate descent.

With the dual variables $\alpha_{ijk} \geq 0$ and $\beta_{ijk} \geq 0$ for the inequality constraints, the Lagrangian of the optimization problem (3.26) is

$$\begin{aligned} &\mathcal{L}(S_i,\xi,\alpha,\beta) = \|S_i-T_i\|_2^2 + \lambda_1\|S_i-U_iV\|_2^2 + \lambda_4 \sum\nolimits_{(i,j,k)\in\mathcal{S}} \xi_{ijk} \\ &- \sum\nolimits_{(i,j,k)\in\mathcal{S}} \big(\alpha_{ijk}(S_{ij} - S_{ik} + \xi_{ijk} - 1) + \beta_{ijk}\xi_{ijk}\big). \end{aligned} \tag{3.27}$$

Taking the derivative of $\mathcal{L}$ with respect to S_i and $\{\xi\}$, we have

$$\frac{\partial \mathcal{L}}{\partial S_i} = 2(S_i - T_i) + 2\lambda_1(S_i - U_iV) - \sum\nolimits_{(i,j,k)} \alpha_{ijk}(e_j - e_k), \tag{3.28}$$

and

$$\frac{\partial \mathcal{L}}{\partial \xi_{ijk}} = \lambda_4 - \alpha_{ijk} - \beta_{ijk}, \quad \forall (i,j,k) \in \mathcal{S}. \tag{3.29}$$

Letting derivatives in equation (3.28) and (3.29) be zero, we have

$$\begin{aligned} S_i^* =& \frac{\sum_{(i,j,k)\in\mathcal{S}} \alpha_{ijk}(e_j - e_k) + 2\lambda_1 U_i V + 2T_i}{2 + 2\lambda_1}, \text{ and} \\ & \alpha_{ijk} + \beta_{ijk} = \lambda_4, \quad \forall (i,j,k) \in \mathcal{S}. \end{aligned} \tag{3.30}$$

We further denote by α_i, β_i, ξ_i the vectorization of α_{ijk}, β_{ijk} and ξ_{ijk} with $(i,j,k) \in \mathcal{S}$ respectively. $\mathcal{R}_i = \{(j,k) : (i,j,k) \in \mathcal{S}\}$ is used to represent indices of the elements of S_i that appear in the constraints, and α_i, β_i, ξ_i are of size $1 \times |\mathcal{R}_i|$. Moreover, we define the matrix M_i of size $|\mathcal{R}_i| \times n$ whose rows are comprised of $\{e_j - e_k, (i,j,k) \in \mathcal{S}\}$, and the rows of M are arranged in the order such that $\alpha_i M_i = \sum_{(i,j,k)\in\mathcal{S}} \alpha_{ijk}(e_j - e_k)$. With these new notations, S_i^* can be rewritten as

$$S_i^* = \frac{\lambda_1 U_i V + T_i}{1 + \lambda_1} + \frac{\alpha_i M_i}{2 + 2\lambda_1}. \tag{3.31}$$

Substituting S_i^* (3.31) into the Lagrangian (3.27), we obtain the dual problem below, which is a box constrained Quadratic Programming (QP) problem:

$$\min_{\alpha_i} \; P(\alpha_i) = \frac{1}{2}\alpha_i Q_i \alpha_i^T - \alpha_i r_i^T \;\; \text{subject to: } 0 \le \alpha_i \le \lambda_4, \tag{3.32}$$

where

$$Q_i = \frac{M_i M_i^T}{2(1+\lambda_1)}, \quad r_i = \mathbf{1} - \frac{(\lambda_1 U_i V + T_i) M_i^T}{1 + \lambda_1}. \tag{3.33}$$

$\mathbf{1}$ is an all-ones $1 \times \mathcal{R}_i$ vector, and the inequality in (3.32) is the element-wise inequality. In addition, according to the KKT conditions, the optimal solution of the primal and dual variables should satisfy:

$$\begin{cases} \alpha_{is}(S_{ij} - S_{ik} + \xi_{ijk} - 1) = 0 \\ \beta_{is}\xi_{is} = 0 \\ \alpha_{is} + \beta_{is} = \lambda_4, \quad \alpha_{is} \ge 0, \; \beta_{is} \ge 0, \end{cases}$$

where α_{is} is the s-th element of α_i and $1 \le s \le |\mathcal{R}_i|$. j, k are the indices that correspond to the s-th constraint. Combined with the primal constraints in

equation (3.26), it follows that

$$\begin{cases} \alpha_{is} = 0 \quad \Rightarrow \; S_{ij} - S_{ik} \geq 1 \\ 0 < \alpha_{is} < \lambda_4 \quad \Rightarrow \; S_{ij} - S_{ik} = 1 \\ \alpha_{is} = \lambda_4, \quad \Rightarrow \; S_{ij} - S_{ik} \leq 1. \end{cases} \tag{3.34}$$

Note that Q_i is positive semi-definite, but it may not be positive definite. Also, it can be verified that the diagonal elements of Q_i are all $\frac{1}{1+\lambda_1}$ since the diagonal elements of $M_i M_i^T$ are all 2. We use coordinate descent method [116] to solve the optimization problem (3.32). In each iteration of the coordinate descent, the objective function $P(\alpha_i)$ is minimized in a coordinate-wise manner. Suppose $\alpha_i^t = \{\alpha_{i1}^t, \alpha_{i2}^t, \ldots, \alpha_{i|\mathcal{R}_i|}^t\}$ is the value of α_i in t-th iteration for $t \geq 0$, the coordinate descent method minimizes α_{is} for $s = 1, 2, \ldots, |\mathcal{R}_i|$ with other elements fixed:

$$\begin{aligned} \alpha_{i1}^{t+1} &= \arg\min_{\alpha_{i1}} P(\alpha_{i1}, \alpha_{i2}^t, \ldots, \alpha_{i|\mathcal{R}_i|}^t) \\ &\ldots \\ \alpha_{is}^{t+1} &= \arg\min_{\alpha_{is}} P(\alpha_{i1}^{t+1}, \alpha_{i2}^{t+1}, \ldots, \alpha_{is}, \alpha_{i(s+1)}^t, \ldots, \alpha_{i|\mathcal{R}_i|}^t) \\ \alpha_{i|\mathcal{R}_i|}^{t+1} &= \arg\min_{\alpha_{i|\mathcal{R}_i|}} P(\alpha_{i1}^{t+1}, \alpha_{i2}^{t+1}, \ldots, \alpha_{i(|\mathcal{R}_i|-1)}^{t+1}, \alpha_{i|\mathcal{R}_i|}). \end{aligned} \tag{3.35}$$

To illustrate the coordinate-wise minimization in (3.35), we show how to optimize over α_{is} with all the remaining elements $\{\alpha_{i1}, \ldots, \alpha_{i(s-1)}, \alpha_{i(s+1)}, \ldots, \alpha_{i|\mathcal{R}_i|}\}$ fixed. In this case, the optimization problem of equation (3.32) is reduced to

$$\min_{\alpha_{is}} \; P(\alpha_{is}) = \frac{1}{2(1+\lambda_1)} \alpha_{is}^2 - R_s \alpha_{is}, \text{ s.t.: } 0 \leq \alpha_{is} \leq \lambda_4, \tag{3.36}$$

where $R_s = r_{is} - \sum_{u \neq s} \alpha_{iu} (Q_i)_{su}$. Equation (3.35) is an univariate QP problem, and $P(\alpha_{is})$ achieves its minimum at

$$\alpha_{is}^* = \begin{cases} \lambda_4 & : \; R_s(1+\lambda_1) > \lambda_4 \\ R_s(1+\lambda_1) & : \; 0 \leq R_s(1+\lambda_1) \leq \lambda_4 \\ 0 & : \; R_s(1+\lambda_1) < 0. \end{cases} \tag{3.37}$$

The coordinate descent algorithm for the dual problem (3.32) for each $1 \leq i \leq n$ is summarized in Algorithm 2.

Algorithm 2: Coordinate Descent Algorithm for the Dual Problem (3.32)

Input: U_i, V, λ_4, $\mathcal{R}_i$: the constraint set, α_i^0: the initial value of α_i, ε_0: the stopping threshold, $\tau_{\max}$: the maximum number of iteration

Output: The i-th row of the similarity matrix S_i

Initialize: $t = 0$, α_i^0 is set to be an all zeros vector. Compute Q_i, r_i according to (3.33).

for $t = 0 \dots \tau_{max} - 1$ **do**

  for $s = 1 \dots |\mathcal{R}_i|$ **do**

    $R_s = r_{is} - \sum_{u \neq s} \alpha_{iu}(Q_i)_{su}$, R_s is computed using $\{\alpha_{i1}^{t+1}, \dots, \alpha_{i(s-1)}^{t+1}, \alpha_{i(s+1)}^{t}, \dots, \alpha_{i|\mathcal{R}_i|}^{t}\}$.

$$\alpha_{is}^{t+1} = \begin{cases} \lambda_4 & : \quad R_s(1+\lambda_1) > \lambda_4 \\ R_s(1+\lambda_1) & : \quad 0 \leq R_s(1+\lambda_1) \leq \lambda_4 \\ 0 & : \quad R_s(1+\lambda_1) < 0 \end{cases}$$

  end

  if $\|\alpha_i^{t+1} - \alpha_i^t\|_2 < \varepsilon_0$ **then**

    The algorithm converges and break

  end

  $t = t + 1$

end

Compute $S_i = \frac{\lambda_1 U_i V + T_i}{1+\lambda_1} + \frac{\alpha_i^* M_i}{2+2\lambda_1}$ using the obtained optimal solution α_i^*.

return S_i

In addition, the dual problem (3.32) has a nice property regarding the number of iterations required to converge. Let P^* denote the minimum value of the objective function for the dual problem (3.32), and $\{\alpha_i^t\}_{t=1}^{\infty}$ be the sequence obtained by the coordinate descent algorithm 2 with $\varepsilon_0 = 0$ and $\tau_{\max} = \infty$. Based on the property of the coordinate descent algorithm [116], algorithm 2 converges and obtains the globally optimal solution to the dual problem (3.32). In fact, since the sequence $\{\alpha_i^t\}_{t=1}^{\infty}$ is bounded, it contains a subsequence that converges to the optimal solution to (3.32) where the optimality condition is met. In practice, the stopping threshold ε_0 is a small positive number and $\tau_{\max}$ is finite. For $\varepsilon_0 > 0$, theorem 3.5.1 gives the upper bound for the number of iterations required for the convergence of algorithm 2. The proof is shown in appendix A.3.

Theorem 3.5.1 *The coordinate descent algorithm 2 converges after at most* $\left\lceil \frac{2|\mathcal{R}_i|(P_0 - P^*)(1+\lambda_1)}{\varepsilon_0^2} \right\rceil$ *iterations, where* $P_0 = P(\alpha_i^0)$ *is the initial value of the*

Table 3.1: The detailed statistics of the datasets.

	DBLP	DBLP-clean	CoRA
Number of node	28,702	2,760	15,644
Number of edge	133,664	7,636	59,062
Number of node with label	4,057	2,760	15,644
Number of class	4	4	10
Content dimensionality	13,214	13,214	12,313

objective function.

We run Algorithm 2 for $i = 1 \dots n$ to obtain the entire S, and the computation of S can be parallelized by computing $\{S_i\}$ separately. Moreover, according to Theorem 2, letting ε_0 be the stopping threshold of the coordinate descent method in Algorithm 2, and $|\mathcal{R}_i|$ be the number of constraints in S_i, Algorithm 2 converges after at most $\left\lceil \frac{2|\mathcal{R}_i|(P_0 - P_i^*)(1+\lambda_1)}{\varepsilon_0^2} \right\rceil$ iterations, where $P_0 = P(\alpha_i^0) = 0$, P_i^* is minimum value of the objective function for the dual problem (32). Therefore, the time complexity for computing S_i is $\mathcal{O}(|\mathcal{R}_i|^2 \left\lceil \frac{2|\mathcal{R}_i||P_i^*|(1+\lambda_1)}{\varepsilon_0^2} \right\rceil + rn)$. Let $\mathcal{R}_{\max} = \max_{1 \le i \le n} \mathcal{R}_i$ be the maximum number of constraints across all the rows of S, $P_{\min}^* = \min_{1 \le i \le n} P_i^*$, then the time complexity for completing the entire S sequentially is $\mathcal{O}(n|\mathcal{R}_{\max}|^2 \left\lceil \frac{2|\mathcal{R}_{\max}||P_{\min}^*|(1+\lambda_1)}{\varepsilon_0^2} \right\rceil + rn^2)$.

3.6 Evaluation

In this section, several experimental results are presented on different datasets in order to validate the effectiveness and efficiency of the proposed FSL method. We also present robustness results in terms of parameter sensitivity and noise tolerance. Our FSL approach on two real datasets and one synthetic dataset significantly outperforms other existing off-the-shelf methods.

3.6.1 Datasets

The detailed descriptions of the datasets are as follows:

DBLP-Four-Areas: DBLP is an online collection of computer science. It is a source of cross-genre information, including content (*e.g.*, keywords of

papers) and links (*e.g.*, co-author relationships, and user friendships). In this chapter, we use the DBLP subset from [37], which contains 28,569 research papers from 28,702 authors, published in 20 conferences. The content information for each paper is extracted from its abstract, and represented using a bag of words. Moreover, 4,057 authors are labeled by four areas, corresponding to database, data mining, information retrieval, and artificial intelligence.

DBLP-Clean: A cleaned version of the DBLP-Four-Areas is also extracted from the original dataset. This cleaned dataset, removing all the authors who do not have, any connection with others or who have no labels, includes 2,760 authors and is labeled by four areas. It is utilized to analyze the performance of the proposed algorithm and verify the robustness on parameter selection.

CoRA: This dataset is comprised of computer science research papers, and includes full citation graph and the topics (and sub-, sub-subtopics) of each paper [117], resulting in over 80 labels. Instead of using such a huge label space, we used the hierarchical structure of the labels provided by the dataset, and used the higher level labels. In our setting, there are 10 group labels, to identify the class of each paper.

Summary statistics of the datasets are illustrated in table 3.1.

3.6.2 Baseline Methods

We compared our proposed method with a number of baseline algorithms including the following:

Euclidean Metric: The standard Euclidean distance between content vectors measures the inverse of the similarity between two nodes.

PMF [57]: Probabilistic Matrix Factorization treats the link matrix L as the utility matrix to complete. PMF only utilizes the existing linkage information as observed entries. The stronger a link between a pair of nodes, the greater the similarity between them.

NMF [85]: Nonnegative Matrix Factorization is similar to PMF in which the link matrix L is used to be completed.

LAD [9]: Locally-Adaptive Decision function learning uses both content

and supervision information to learn a local non-isotropic similarity function beyond the traditional generalized Mahalanobis metric.

CFSL: Content-based Factorized Similarity Learning is a special case of our FSL algorithm by setting $\lambda_4 = 0$ in Equation (3.22). CFSL is still able to incorporate both link and content information in a globally factorized manner.

SSMetric [**8**]: Semi-Supervised Metric learning incorporates knowledge from sparse linkage information and is used as neighborhood graph. It is a variant of the originally proposed method, which is modified to allow it to use the linkage structure. The intentional knowledge can be propagated through the link graph L to learn a distance metric on the content vector space.

In summary, the first two baselines learn a similarity measure based only on content or linkage information in an unsupervised manner. LAD utilizes both content and supervised knowledge. CFSL evaluates the proximity on both contents and links. SSMetric is similar to our method in terms of incorporating different information sources on content, linkages and supervision.

3.6.3 Experimental Settings

In our experiments, we simulated the real-world scenario on similarity learning as a retrieval problem. We start by explaining the experimental settings with an example. As illustrated in figure 3.3, we divide all pairwise nodes into two disjoint groups parameterized by two variables p_v and p_h indicating the level of supervision. For instance, if $p_h = 0.5$ and $p_v = 0.6$, then it means $0.5 + 0.6 \times (1 - 0.5) = 80\%$ of entries are provided supervised knowledge, and the remaining 20% do not have any information about relative ordering. It is worth mentioning that if we divide the training and testing portions into portions of size 80% and 20%, it does not mean that the full triplet constraints will be given for the training region. Another hyper-parameter s controls the number of triplet orderings provided for the training region. In our experiments, s is usually set to the range of 5 to 20.

Since the ground truth provided in both the DBLP and CoRA datasets are explicit multi-class labels, we need to convert them into triplet constraints.

One way of achieving this is to generate triplet constraints, is by setting nodes with the same label as a similar pair and nodes with different labels as dissimilar. In other words, the triplet constraint $(i, j, k) \in \mathcal{S}$ is generated by randomly choosing two nodes v_i and v_j with the same label. And v_k has a different label with v_i and v_j.

The implementations of LAD and SSMetric methods use pairwise constraints instead of triplets. Although straightforward conversions exist from pairwise settings to triplet in most metric learning based algorithms, we obey their original implementation by converting triplet constraint to pairwise in the following way: each triplet constraint (i, j, k) is split into two different sets, that is, (v_i, v_j) as a similar pair and (v_i, v_k) as a dissimilar pair. Another issue for these two baselines is that they are not able to scale up to a high dimensional setting. Therefore, we perform Principal Component Analysis (PCA) to reduce the dimensionality to $1,000$ as a preprocessing step.

For each dataset, we initialize our similarity matrix S by the link matrix L with a small constant value to each entry. The purpose of adding a small constant value in S is to prevent a row or a column of S without any initial value. Adding a constant value to every entry of the similarity matrix will not affect the performance, since we only emphasize the ordered information instead of the explicit entrywise values. Similar initialization is conducted on the bridging matrix T as well. To initialize the low-rank matrix U, V and W we use a Laplace distribution with zero mean and a scale parameter value of one. In addition, the content matrix C and the link matrix L are normalized to remove the scale variations. To evaluate the performance, we compute the averaged precision at each level k (denoted as $P@k$) across different query nodes.

3.6.4 Results

In this section, we present the results from our proposed FSL approach and the aforementioned baseline methods on both DBLP and CoRA datasets. All experimental results were averaged over 10 runs.

DBLP:

According to our experimental settings, we provide each node 30 triplet con-

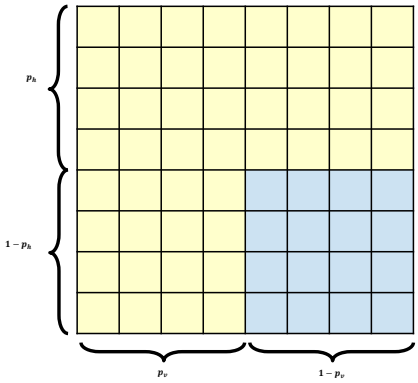


Figure 3.3: The experiment settings; yellow region indicates the training while blue is the testing entries.

straints as the intentional knowledge and report the comparative performance with other baseline methods in figure 3.4. It is evident that the proposed method achieves the best performance across all ranges of the ranks tested. On the other hand, link-based methods such as PMF and NMF achieved the poorest performance. The other methods achieved intermediate performance. The LAD method achieves the second best performance for learning similarity between authors in the publication network.

An interesting observation is that all methods using linkage information performed worse than the content-based methods, except for the proposed FSL scheme. The reason for this is that the noisy links can often hurt the proximity approximation. Predictions from PMF and NMF methods are based only on the sparse noisy links without any global content bias. CFSL utilizes both content and linkage information. However, the noise encoded in the linkage structure prevents good prediction results. SSMetric is similar to the proposed FSL method which uses linkage, content and supervision simultaneously. However, it is particularly poor at handling noise because of its inability to prevent similarity propagation along noisy links.

The LAD algorithm incorporates the supervised information to learn semantic proximities, which outperform unsupervised content methods. However, the useful information within the linkage structure cannot be utilized to enhance the performance. The proposed FSL approach is able to identify these unreliable links and eliminate their contributions by transferring and fusing the knowledge from content and supervision. In such a way, influential links can be emphasized, so that FSL achieves the best performance.

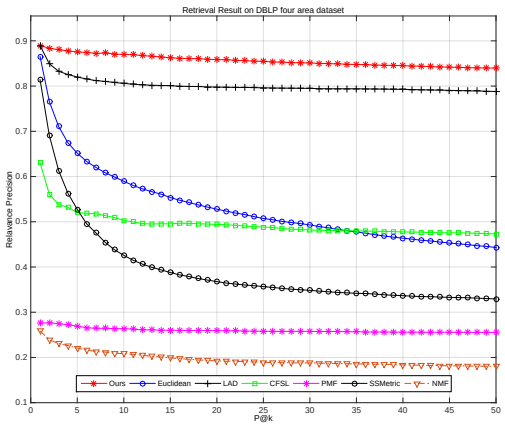


Figure 3.4: P@k curve on the DBLP dataset.

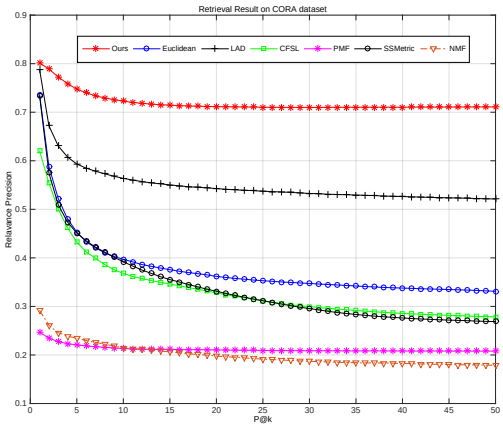


Figure 3.5: P@k curve on the CoRA dataset.

CoRA:

Since the CoRA dataset is somewhat smaller than DBLP in terms of the number of nodes and links, we only provided 15 supervised examples per node. We reported the top 50 retrieval results for each baseline method in Figure 3.5. We obtain similar results to the DBLP dataset, on which the linkage-based method performed poorly. The PMF and NMF methods obtain the worst result. Although the performance of CFSL and SSMetric is comparable with the standard Euclidean metric, they are still not quite in the same league as the LAD approach.

The proposed method outperforms LAD by more than 10%, starting from rank 5, and retains this performance beyond this point. It shows that the proposed FSL method not only estimates the proximity of top candidates correctly, but also retains a very high recall in the retrieval tasks. Our proposed method is very robust in terms of the similarity learning across different datasets.

Discussion:

Comparing the experimental results we obtained from figure 3.4 and figure 3.5, we discover that the precision decreases much slower with k increases for the DBLP dataset. Specifically, the precision of our proposed method at 50 for the DBLP dataset still remains around 0.85 while the CoRA dataset only has 0.7 left. Similar observations are also reflected from other baselines.

This is due to the number of classes for the CoRA dataset being significantly larger than in the DBLP dataset. In addition, the labels' granularity is much finer for the CoRA dataset, which imposes huge challenges to correctly retrieve other "similar" nodes. Although the absolute precision of our model for the CoRA dataset is lower than that for the DBLP dataset, the relative performance of our algorithm compared to the second best method is much better. This implies that our performance drops much slower compared to other baselines when the retrieval task gets much harder.

3.6.5 Parameter Sensitivity

The main parameters of the proposed FSL algorithm are the weight parameters λ_i, the portion of supervision information s (the number of constraints provided in training for each user), and the rank of matrices U and V (denoted as R). To validate the robustness of parameters and analyze the effect of each parameter on the final result, a group of experiments were conducted on the clean DBLP dataset. It is a small dataset, obtained by cleaning all the noise from DBLP, and contains links, content and four classes. We use the strategy in Section 3.6.3 to generate supervision information.

Control Parameters λ_i:
The performance with varying λ_1 is shown in figure 3.6, in which λ_2 is fixed at 7, $R = 10$ and $s = 12$. λ_1 controls the importance of linkage information considered in factorization. As shown in figure 3.6, the performance is stable when $\lambda_1 \geq 1$. The results suggest that as long as sufficient linkage information is provided, the content similarity and supervision can be robustly propagated along the topological structure.

Similarly, the effect of λ_2 is shown in figure 3.7, and the performance is robust to parameter setting when $\lambda_2 > 3$. It validates the importance of global (content) information on similarity learning. The robustness in parameter choice reflects how optimality is achieved with the help of underlying topological structure spread with linkage information.

A comparison between figures 3.6 and 3.7, yields some interesting observations:

- when λ_1 increases, the performance drops slightly;

- when λ_2 increases, the performance improves slightly.

This observation is in agreement with our experimental results in Section 3.6.4. For this particular task assignment, linkage information is not as useful as content similarity.

Control Parameters s:
Figure 3.8 shows the effect of supervision on the FSL algorithm, fixing $\lambda_1 = 1.5$, $\lambda_2 = 7$ and $R = 10$. It is obvious that given a certain number of constraints for each user, *i.e.* $s > 10$, the performance is fairly stable regardless of the value of s. These results suggest the following:
s **increases**: As more supervision is provided, the FSL algorithm will adjust the topological structure of networks relying on trustworthy guidance. In this situation, the information propagation will be more efficient. On the other hand, diminishing returns are achieved for increasing s beyond a certain point.
s **is small**: In this case, the algorithm focuses most of its efforts on fitting a small portion of supervision. This has a detrimental impact on the whole structure of the network. As a result, the performance is not very good in this range.

In this experiment, the percentage of supervision is $p_s = s/N(U)$, which is approximately 4×10^{-4}. This is much smaller than a typical social network, *e.g.*, Facebook, where there are hundreds of labeled links (*i.e.*, friendships) on average for each user. Therefore, the algorithm is practical in real-world scenarios.

Low Rank Approximation R:
Finally, the effect of matrix rank R is shown in figure 3.9. As observed in the figure, the performance increases stably after $R \geq 8$. Considering the fact that the samples in the DBLP dataset are labeled with 4 classes, it is feasible to assume $R > 4$. Typically, the value assignment of rank R is application-dependent.

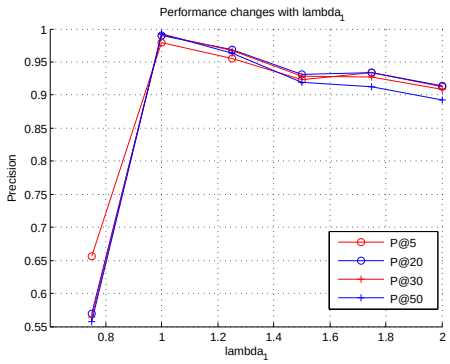


Figure 3.6: Parameter testing: λ_1.

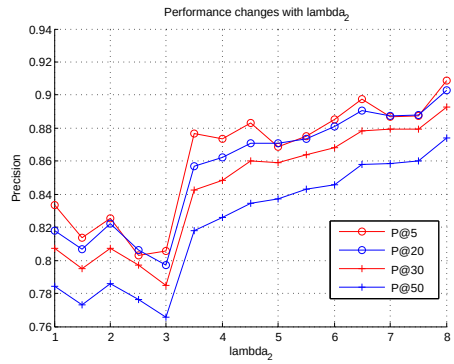


Figure 3.7: Parameter testing: λ_2.

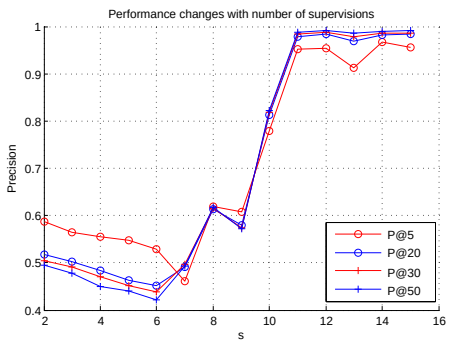


Figure 3.8: Parameter testing: number of supervision - s.

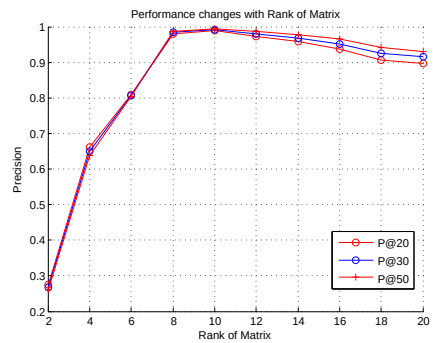


Figure 3.9: Parameter testing: number of supervision - R.

3.6.6 Noise Tolerance

In this section, we present the performance on error tolerance using the large-margin formulation proposed in equation (3.21) on the DBLP-clean dataset. We tested the FSL method with different levels of noise in the supervision in figure 3.10. The color of the histogram indicates the level of noise injection. Furthermore, the different groups in the histogram show the retrieval result at different ranks. We observe that when the noise level is low (1% or 5%) the proposed method maintains very good results, and the retrieval precision decreases very slowly with increasing rank. However, when the noise level becomes high, the FSL method obtains a poor recall. Overall, figure 3.10 demonstrates that our proposed method is robust to a low-level of error tolerance.

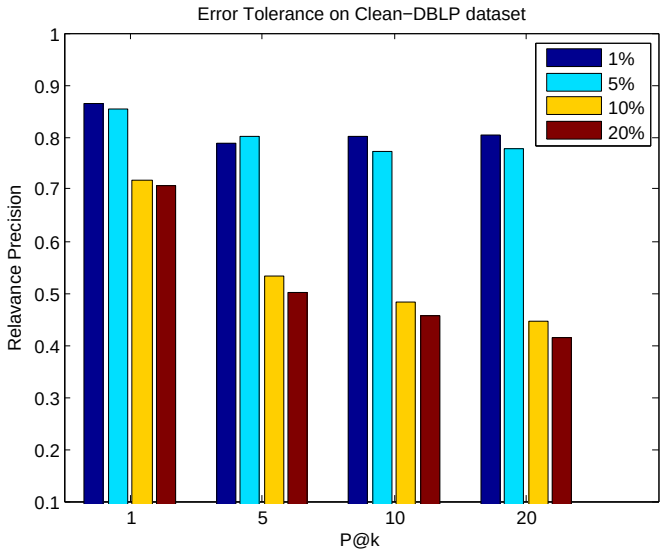


Figure 3.10: Error tolerance: different color indicates the percentage of supervision randomly flipped.

3.6.7 Efficient Solution by the Dual

Directly solving the primal problem (3.23) needs to handle n^2 elements of S by the subgradient method. In contrast, the efficient dual solver only deals with $|\mathcal{R}_i|$ variables for each row of S, with a total of $n|\mathcal{R}_i|$ variables, and it leads to a much more efficient solution. We perform the comparison of computational time between the optimization of (3.23) in the primal form using subgradient versus the dual form using quadratic programming by coordinate descent. Figure 3.11 shows the comparison of the computational time using fixed number of users $n = 10^4$, with the number of constraints varies within $\{1, 100, 200, \ldots, 1000\}$. It is observed that the dual method always needs less time than the primal method. In addition, both of them take more time with the increasing number of constraints. With more constraints, more computational cost arises when computing the subgradient for the primal method, and there are more variables in the dual method. Figure 3.12 illustrates the comparison of the computational time using fixed number of constraints, *i.e.* $|\mathcal{R}_i| = 100$ for all rows of S, with the number of users varies within $\{10^4, 10^5, 2 \times 10^5, \ldots, 10^6\}$. In this case, the number of variables for the dual method is fixed, and the number of variables for the primal method increases quadratically with the number of users. We can see that the dual method is significantly faster than the primal method. Also, the computational time of

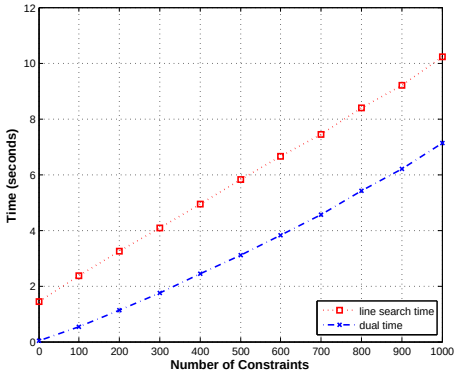


Figure 3.11: Comparison of Computation Time with varying number of constraints and fixed number of users.

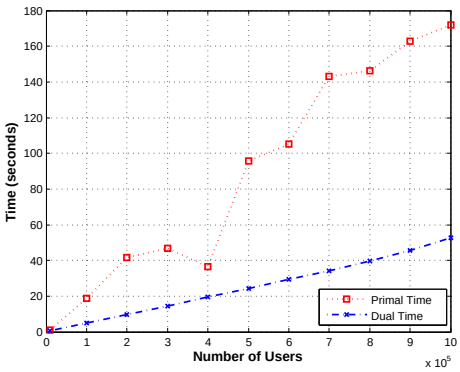


Figure 3.12: Comparison of Computation Time with varying number of users and fixed number of constraints.

the dual method increases with more and more users, since the dual method still needs to compute Q_i, r_i and S_i for each row of S.

Note that the rows of S can be computed separately. In both comparisons, the first 300 rows of S are computed, and the Frobenius norm of the difference of S computed using the primal and the dual is always less than 10^{-7}. The maximum number of iterations for the subgradient method in the primal and the coordinate descent in the dual is 200. We perform the comparisons on a Desktop with 16 GB memory and Intel i7-4770 3.4 GHz CPU.

3.7 Conclusion

In this chapter, we proposed a novel similarity learning approach, termed as FSL, to measure node-based similarity in large networks within a matrix factorization framework. We propose a holistic model, which leverages network topological structure, node content and user supervision. The proposed method is able to ameliorate the impact of noisy linkage structures by fusing and transferring knowledge from other domains. At the same time, the reliable linkages are used effectively in conjunction with content and user-supervision. By embedding content and links into a unified latent space, the supervision can correctly guide the factorization process. We show extensive experiments on real-world datasets. The proposed FSL method significantly outperforms other state-of-the art approaches in node-based retrieval, and is efficient and highly robust for noisy supervision.

CHAPTER 4

VARIED DATA

In the previous chapter, we illustrated how supervised information is extremely useful in distinguishing the notion of "similarity" in networks from its dual aspects. The content associated with each node is explicitly considered as homogeneous, which means it belongs to a single modality. However, "variety" is one of the three-V characteristics of big data; it is more realistic to assume the inputs for real-world systems consist of data from different domains.

In this chapter, we further investigate the problem of similarity learning in more depth by examining heterogeneous networks, where the content and nodes are of various types. Such networks are notoriously difficult to mine because of the bewildering combination of the heterogeneous content and structure. We aim to alleviate the "variety" challenge posed by big data by creating a unified embedding framework for heterogeneous networks, which converts each network node into a multidimensional representation in an unsupervised fashion. In other words, a desired embedding scheme serves as a feature learning process to transform every node to a vectorized representation by encoding both content and link similarity from networks. The idea of a network-preserved embedding is illustrated in figure 4.1. The creation of the network embedding opens the door to the use of a wide variety of off-the-shelf learning techniques for multidimensional data.

4.1 Heterogeneous Network Representation

A heterogeneous network [60] is defined as a network with multiple types of objects and/or multiple types of links. As a mathematical abstraction, we define an undirected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ is a set of vertices and $\mathcal{E}$ is a set of edges. An edge $e_{ij},\ \forall i, j \in \{1, \cdots, n\}$ belongs

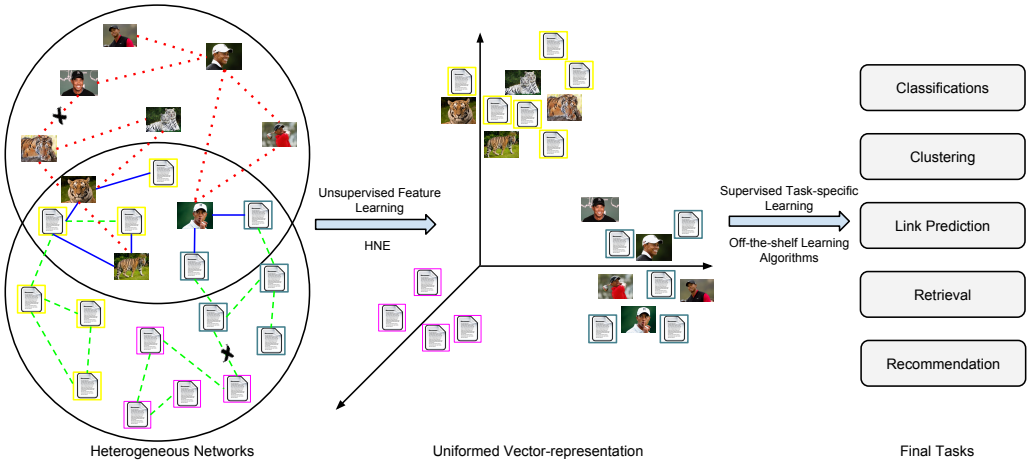


Figure 4.1: The flowchart of the proposed Heterogeneous Network Embedding (HNE) framework.

to the set $\mathcal{E}$ if and only if an undirected link exists between nodes i and j. Moreover, the graph G is also associated with an object type mapping function $f_v : \mathcal{V} \rightarrow \mathcal{O}$ and a link type mapping function $f_e : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{O}$ and $\mathcal{R}$ represent the object and relation sets, respectively. Each node $v_i \in \mathcal{V}$ belongs to one particular object type as $f_v(v_i) \in \mathcal{O}$. Similarly, each link $e_{ij} \in \mathcal{E}$ is categorized in different relations as $f_v(e_{ij}) \in \mathcal{R}$. It is worth mentioning that the linkage type of an edge automatically defines the node types of its end points. The heterogeneity of a network is reflected by the size of the sets $\mathcal{O}$ and $\mathcal{R}$, respectively. In the case of $|\mathcal{O}| = |\mathcal{R}| = 1$, the network is homogeneous; otherwise, it is heterogeneous. An example of a heterogeneous network is illustrated in the left-hand side of figure 4.1, which contains two object and three link types. For further ease in understanding, we will assume object types of image (I) and text (T). The link relationships $\mathcal{R}$ correspond to image-to-image (red dotted line), text-to-text (green dashed line) and image-to-text (blue solid line), which are denoted by R_{II}, R_{TT} and R_{IT}, respectively. Therefore, in this case, we have $|\mathcal{O}| = 2$ and $|\mathcal{R}| = 3$. While this simplified abstraction in the text and image domain is both semantically and notationally convenient for further discussion in this chapter, this assumption is without loss of generality because the ideas are easily generalizable to any number of types.

Thus, any vertex $v_i \in \mathcal{V}$ can be categorized into two disjoint subsets $\mathcal{V}_I$ and $\mathcal{V}_T$ corresponding to the text image domains, respectively. Therefore, we have

$\mathcal{V}_I \cup \mathcal{V}_T = \mathcal{V}$ and $\mathcal{V}_I \cap \mathcal{V}_T = \phi$. Similarly, the edge set $\mathcal{E}$ can be partitioned into three disjoint subsets, which are denoted by $\mathcal{E}_{II}$, $\mathcal{E}_{TT}$ and $\mathcal{E}_{IT}$, respectively. Furthermore, each node is summarized by unique content information. In particular, images are given as a squared tensor format as $X_i \in \mathbb{R}^{d_I \times d_I \times 3}$ for every $v_i \in \mathcal{V}_I$, while texts are represented by a d_T-dimensional feature vector as $z_j \in \mathbb{R}^{d_T}$ for all $v_j \in \mathcal{V}_T$. For example, the content representation could be a raw pixel format in RGB color space for images, or it could be the Term Frequency - Inverse Document Frequency (TF-IDF) [10] scores of a text document. We represent linkage relationship as a symmetric matrix $L \in \mathbb{R}^{n \times n}$, in which the (i, j)th entry of L equals one if $e_{ij} \in \mathcal{E}$; otherwise $L_{ij} = -1$ (for model simplicity).

4.2 Heterogeneous Network Embedding

In this section, we present Heterogeneous Network Embedding (HNE) mathematically by first introducing a novel loss function to measure correlations across networks. Essentially, the embedding process encodes both heterogeneous content and linkage information to a multidimensional representation for each object.

4.2.1 Latent Embedding in Networks

The main goal of the heterogeneous embedding task is to learn a mapping function to project data from different modalities to a common space so that similarities between objects can be directly measured. Assume that the raw content X_i associated with an image node can be transformed to a d'_I-dimensional vector representation as x_i. The conversion of the raw input data into this d'_I-dimensional vector representation can be achieved by using any feature machines. A naive approach to do so is by stacking each column of an image as a vector or through feature machines. It is worth pointing out that the values of d'_I and d_T need not be the same, because images and text are defined in terms of completely different sets of features.

We transform two types of samples to a uniform latent space with the use of two linear transformation matrices, denoted by $U \in \mathbb{R}^{d'_I \times r}$ and $V \in \mathbb{R}^{d_T \times r}$, for the image and text domains, respectively. The transformed samples are

denoted by $\tilde{x}$ and $\tilde{z}$ for images and text documents, respectively, where we have:

$$\tilde{x} = U^T x, \text{ and } \tilde{z} = V^T z. \tag{4.1}$$

Even though the image and documents may be represented in spaces of different dimensionality, the transformation matrices U and V map them into a common r-dimensional space. The similarity between two data points with the same object type can be presented as an inner product in the projected space as follows:

$$\begin{aligned} s(x_i, x_j) &= \tilde{x_i}^T \tilde{x_j} = (U^T x_i)^T U^T x_j = x_i^T M_{II} x_j, \\ s(z_i, z_j) &= \tilde{z_i}^T \tilde{z_j} = (V^T z_i)^T V^T z_j = z_i^T M_{TT} z_j. \end{aligned} \tag{4.2}$$

Note that the embedding into a common space also enables similarity computation between two objects of different types, such as text and images, as follows:

$$\begin{aligned} s(x_i, z_j) &= \tilde{x_i}^T \tilde{z_j} = (U^T x_i)^T V^T z_j = x_i^T M_{IT} z_j \\ &= \tilde{z_j}^T \tilde{x_i} = (V^T z_j)^T U^T x_i = z_j^T M_{IT}^T x_i. \end{aligned} \tag{4.3}$$

Here, $M_{II} \in \mathbb{R}^{d'_I \times d'_I}$ and $M_{TT} \in \mathbb{R}^{d_T \times d_T}$ are positive semi-definite matrices while $M_{IT} \in \mathbb{R}^{d'_I \times d_T}$. The latent embedding is closely related to similarity and metric learning that has been widely studied in the literature [9]. It suggests that the correlations between two nodes in a network can be either parameterized by the projection matrices U and V or through a bilinear function defined by the matrices M_{II}, M_{TT} and M_{IT}. This provides the flexibility to model the heterogeneous relationship in an application-specific way.

The heterogeneous objects interact with each other either explicitly or implicitly. These interactive pieces of information are represented as heterogeneous linkages in networks. The assumption is that if two objects are connected, the similarity measure between them should reflect this fact by providing a larger value compared to the ones that are isolated. Consider a pair of images denoted as x_i and x_j. To encode the link information, we design a pairwise decision function $d(x_i, x_j)$ as follows:

$$d(x_i, x_j) \begin{cases} > 0 & \text{if } L_{ij} = 1, \\ < 0 & \text{otherwise.} \end{cases} \tag{4.4}$$

Note that to infer d we need not know the respective entry value of L, or require heterogeneous nodes to be in-sample. This means that the approach has the generalization ability to embed samples from unseen nodes. Consider

$$d(x_i, x_j) = s(x_i, x_j) - t_{II}, \tag{4.5}$$

for all $v_i, v_j \in \mathcal{V}_I$, where t_{II} is a relational based bias value. Then, the loss function can be formulated as follows:

$$L(x_i, x_j) = \log\left(1 + \exp\left(-L_{i,j} d(x_i, x_j)\right)\right), \tag{4.6}$$

which can be seen as a binary logistic regression guided by network linkages. The loss function of text-text and image-text are similar to equation (4.6) by simply replacing $s(\cdot)$ with that of the corresponding modality. Similarly, the bias terms, denoted by t_{TT} and t_{IT}, can be set to the corresponding ones. It leads to our objective functions in the form of:

$$\begin{aligned} \min_{U,V} \quad & \frac{1}{N_{II}} \sum_{v_i, v_j \in \mathcal{V}_I} L(x_i, x_j) + \frac{\lambda_1}{N_{TT}} \sum_{v_i, v_j \in \mathcal{V}_T} L(z_i, z_j) \\ & + \frac{\lambda_2}{N_{IT}} \sum_{v_i \in \mathcal{V}_I, v_j \in \mathcal{V}_T} L(x_i, z_j) + \lambda_3 (\|U\|_F^2 + \|V\|_F^2), \end{aligned} \tag{4.7}$$

where N_{II}, N_{TT} and N_{IT} are the numbers of the three types of links in the network. Furthermore, λ_1, λ_2 and λ_3 are the three balancing parameters, in which the first two control the emphasis among three types of linkages and the last one is used to balance the so called bias-variance trade-off. The bias terms in the loss functions can be either treated as learning variables or set to fixed values. For simplicity, we set these bias terms to constants.

The aforementioned objective function can be efficiently solved with coordinate descent methods, which solve for each individual variable while keeping the others fixed.

Solving U:

Fixing parameter V, the objective function (4.7) can be reduced as follows:

$$\begin{aligned}\min_{U} \quad & \frac{1}{N_{II}} \sum_{v_i, v_j \in \mathcal{V}_I} \log\left(1 + e^{-L_{i,j} x_i^T U U^T x_j}\right) + \lambda_3 \|U\|_F^2 \\ & + \frac{\lambda_2}{N_{IT}} \sum_{v_i \in \mathcal{V}_I, v_j \in \mathcal{V}_T} \log\left(1 + e^{-L_{i,j} x_i^T U V^T z_j}\right).\end{aligned} \tag{4.8}$$

The gradient is given by the following:

$$\begin{aligned}\frac{\partial(\cdot)}{\partial U} = & \frac{1}{N_{II}} \sum_{v_i, v_j \in \mathcal{V}_I} \frac{-L_{i,j}(x_j x_i^T + x_i x_j^T) U}{1 + e^{L_{i,j} x_i^T U U^T x_j}} + 2\lambda_3 U \\ & + \frac{\lambda_2}{N_{IT}} \sum_{v_i \in \mathcal{V}_I, v_j \in \mathcal{V}_T} \frac{-L_{i,j} x_i z_j^T V}{1 + e^{L_{i,j} x_i^T U V^T z_j}}.\end{aligned} \tag{4.9}$$

Solving V:

Similarly, the variable V can be handled as follows:

$$\begin{aligned}\min_{V} \quad & \frac{\lambda_1}{N_{TT}} \sum_{v_i, v_j \in \mathcal{V}_T} \log\left(1 + e^{-L_{i,j} z_i^T V V^T z_j}\right) + \lambda_3 \|V\|_F^2 \\ & + \frac{\lambda_2}{N_{IT}} \sum_{v_i \in \mathcal{V}_I, v_j \in \mathcal{V}_T} \log\left(1 + e^{-L_{i,j} x_i^T U V^T z_j}\right).\end{aligned} \tag{4.10}$$

Taking the derivative with respect to V, we obtain:

$$\begin{aligned}\frac{\partial(\cdot)}{\partial V} = & \frac{\lambda_1}{N_{TT}} \sum_{v_i, v_j \in \mathcal{V}_T} \frac{-L_{i,j}(z_j z_i^T + z_i z_j^T) V}{1 + e^{L_{i,j} z_i^T V V^T z_j}} + 2\lambda_3 V \\ & + \frac{\lambda_2}{N_{IT}} \sum_{v_i \in \mathcal{V}_I, v_j \in \mathcal{V}_T} \frac{-L_{i,j} z_j x_i^T U}{1 + e^{L_{i,j} x_i^T U V^T z_j}}.\end{aligned} \tag{4.11}$$

So far, we have shown that our loss function integrates network structures that map different heterogeneous components into a unified latent space. However, such embedding functions are still linear, which might lack the power to model complex network connections. In the following, we will present how Equation (4.7) fits into the deep learning framework.

4.3 A Deep Embedding

The previous section, we broke the learning down into two steps: 1) manually construct a feature representation, 2) embed different modalities into a common space. In this section we tightly integrate these two steps into a deep learning framework by learning the feature representation and embedding together:

$$\min_{U,V,\mathcal{D}_I,\mathcal{D}_T} \frac{1}{N_{II}} \sum_{v_i,v_j \in \mathcal{V}_I} L(p_{\mathcal{D}_I}(X_i), p_{\mathcal{D}_I}(X_j)) + \lambda_3(\|U\|_F^2 + \|V\|_F^2)$$
$$+\frac{\lambda_1}{N_{TT}} \sum_{v_i,v_j \in \mathcal{V}_T} L(q_{\mathcal{D}_T}(z_i), q_{\mathcal{D}_T}(z_j)) + \frac{\lambda_2}{N_{IT}} \sum_{v_i \in \mathcal{V}_I, v_j \in \mathcal{V}_T} L(p_{\mathcal{D}_I}(X_i), q_{\mathcal{D}_T}(z_j)). \tag{4.12}$$

Here, $p(\cdot)$ and $q(\cdot)$ are two nonlinear functions parameterized by $\mathcal{D}_I$ and $\mathcal{D}_T$. $\mathcal{D}_I$ and $\mathcal{D}_T$ are two sets of parameters associated with the deep image and text networks, respectively. Specifically, we utilize the convolutional structure as building blocks to learn image features while fully connected (FC) layers are used to extract discriminative representations for pre-processed texts. The feature learning and information embedding are mutually reinforced by our approach.

The image module exploits spatially local correlations by enforcing a local connectivity between neurons from adjacent layers. The parameters on each layer are referred to as filters. The architecture confines the learned filters to reflect the spatial local patterns of images. In addition, each sparse filter is replicated across the entire visual field, which shares the same parameters (both weights W_I^k and bias b_I^k). The output of each filter is usually termed as a "feature map," and conceptually, a feature map is obtained by convolving an input image with a linear filter, adding a bias term and then applying a non-linear function. We denote the k-th feature map at a given layer (a given depth) as h^k, which is determined by the corresponding weights W^k and bias b^k. Then, the feature map is obtained as follows:

$$h^k = \max\{0, (W^k * M) + b^k\}. \tag{4.13}$$

Here, $*$ denotes the convolution operation and M is an input from the previous layer of the deep image module. The definition of convolution of a filter

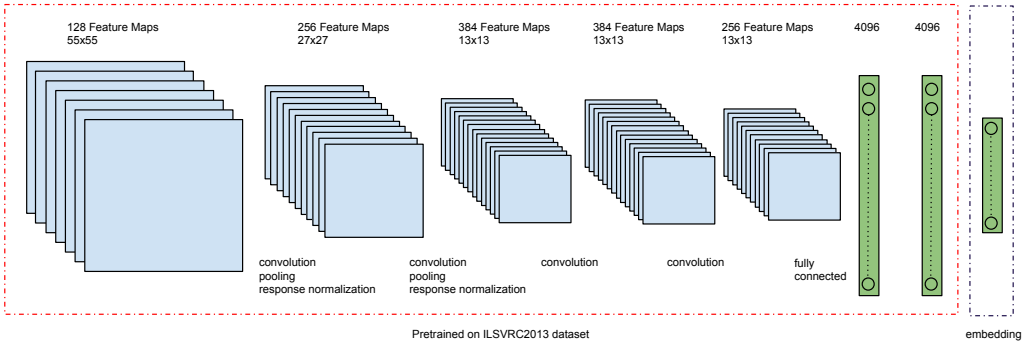


Figure 4.2: An example of the deep image module which consists of five convolution layers and two fully connected layers.

g with a $2D$ signal f is as follows:

$$o[m,n] = f[m,n] * g[m,n] = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} f[u,v]g[u-m,v-n]. \tag{4.14}$$

Moreover, the $\max\{0,\cdot\}$ operator, called the ReLU [94], provides the non-linearity. To form rich representations of a given dataset, each layer is composed of multiple feature maps so that each filter W^k forms a three-dimensional tensor for every combination of source feature map, vertical and horizontal size. A graphical illustration of the image module is provided in figure 4.2, which contains five convolution layers and two FC layers. Each input image $X \in \mathbb{R}^{d_I \times d_i \times 3}$ is represented as a 4096-dimensional vector through a series of nonlinear operations in both the training and testing phases. Once the set of parameters $\mathcal{D}_I$ is fixed, the feature of each individual input image is deterministic.

In contrast, since text documents are unstructured and do not contain spatial information, fully connected layers are commonly used to extract application orientated features on top of TF-IDF inputs. The feature transformation is expressed as follows:

$$q_{\mathcal{D}_T}(z) = \max\{0, W_T z + b_T\}. \tag{4.15}$$

This is performed through a single fully connected layer, where $W_T \in \mathbb{R}^{r \times d_T}$ and $b_T \in \mathbb{R}$. The term r indicates the number of neurons in a given layer. Similarly, rich representations can be learned by stacking multiple fully con-

nected layers with different number of neurons (r can be set to different values in different layers) to construct the deep text architecture for word documents.

Since the linear heterogeneous embedding in section 4.2.1 can be viewed as transforming inputs to a common space, we can achieve this by cascading an extra linear embedding layer to each deep module. Define

$$\tilde{p}_{\mathcal{D}'_I}(X) = U^T p_{\mathcal{D}_I}(X), \text{ and } \tilde{q}_{\mathcal{D}'_T}(z) = V^T p_{\mathcal{D}_T}(z), \tag{4.16}$$

where $\mathcal{D}'_I = \mathcal{D}_I \cup \{U\}$ and $\mathcal{D}'_T = \mathcal{D}_T \cup \{V\}$. Then, the objective function in equation (4.12) is equivalent to the following:

$$\begin{aligned}\min_{\mathcal{D}'_I, \mathcal{D}'_T} \quad & \frac{1}{N_{II}} \sum_{v_i, v_j \in \mathcal{V}_I} L'(\tilde{p}_{\mathcal{D}'_I}(X_i), \tilde{p}_{\mathcal{D}'_I}(X_j)) + \frac{\lambda_1}{N_{TT}} \sum_{v_i, v_j \in \mathcal{V}_T} L'(\tilde{q}_{\mathcal{D}'_T}(z_i), \tilde{q}_{\mathcal{D}'_T}(z_j)) \\ & + \frac{\lambda_2}{N_{IT}} \sum_{v_i \in \mathcal{V}_I, v_j \in \mathcal{V}_T} L'(\tilde{p}_{\mathcal{D}'_I}(X_i), \tilde{q}_{\mathcal{D}'_T}(z_j)).\end{aligned} \tag{4.17}$$

The problem of over-fitting can be effectively prevented by using Dropout [93] instead of L_2 regularizations. The new loss term $L'(\cdot, \cdot)$ is defined as

$$L'(a, b) = \log\left(1 + \exp\left(-A_{i,j} a^T b\right)\right), \tag{4.18}$$

for any vector a, b with a same dimensionality. For simplicity, we refer to both deep image and text modules as a series of nonlinear feature transformations with an additional linear common space embedding.

To perform end-to-end HNE learning, we connect the deep image and text modules accordingly to the image-image, text-text, and image-text losses in equation (4.17). As an example, we illustrate the text-text module in figure 4.3, and the other two can be extended in a similar manner. Figure 4.3 contains two text modules that comprise the pairwise text-text module. The illustrated deep text-text module contains two FC layers followed by a linear embedding layer. A pair of text documents is fed from the left and computed in a left-to-right direction. The outputs from the embedding layer are the vectorized representation of corresponding objects in the common latent space. These are further channeled to a prediction layer to calculate the loss using equation (4.17). To make the text-text module symmetric (feeding the same objects from the top or the bottom pass of the text-text

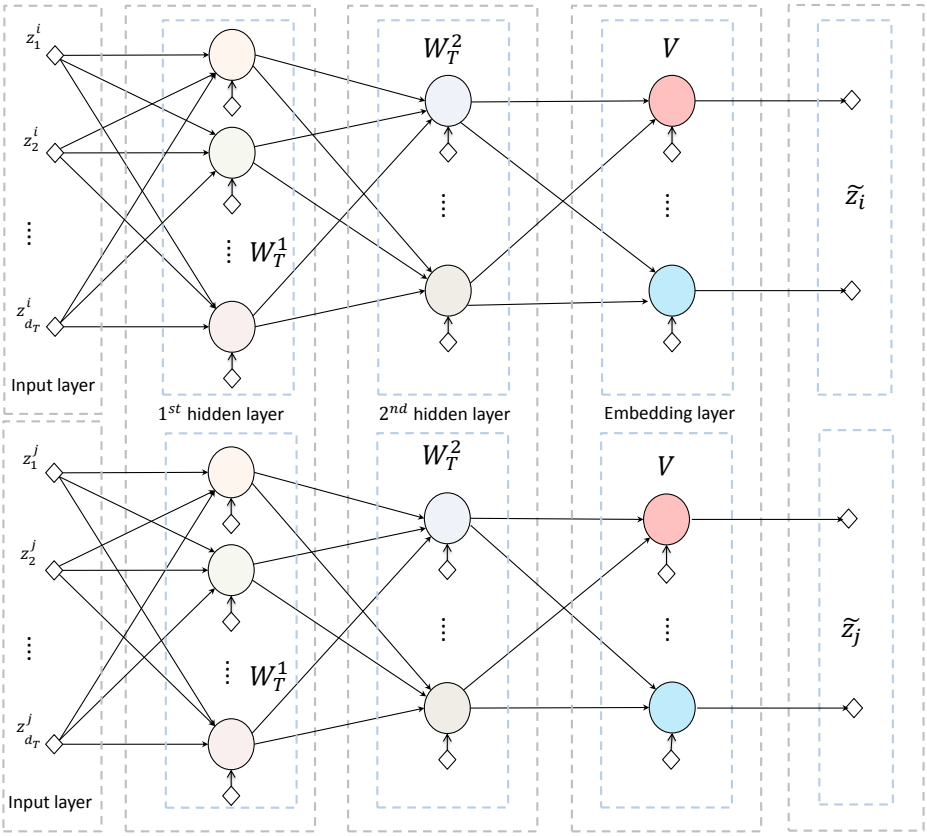


Figure 4.3: An example of the deep text-text module by concatenating a pair of text modules. Same coloring indicates shared weights.

modules will lead to a same latent representation), we need to tighten these parameters. In figure 4.3, if two neurons have the same color, they share the same weight and bias.

The overall architecture of learning such a heterogeneous embedding function from a given network is visualized in figure 4.4. Three modules are shown in the figure, corresponding to image-image, image-text and text-text from left to right. These are connected to the prediction layer. Pairwise training samples are formed as mini-batches feeding from the bottom to the top. Once the value of the loss has been obtained, the gradients of each parameter in the deep network are calculated using backpropagation techniques.

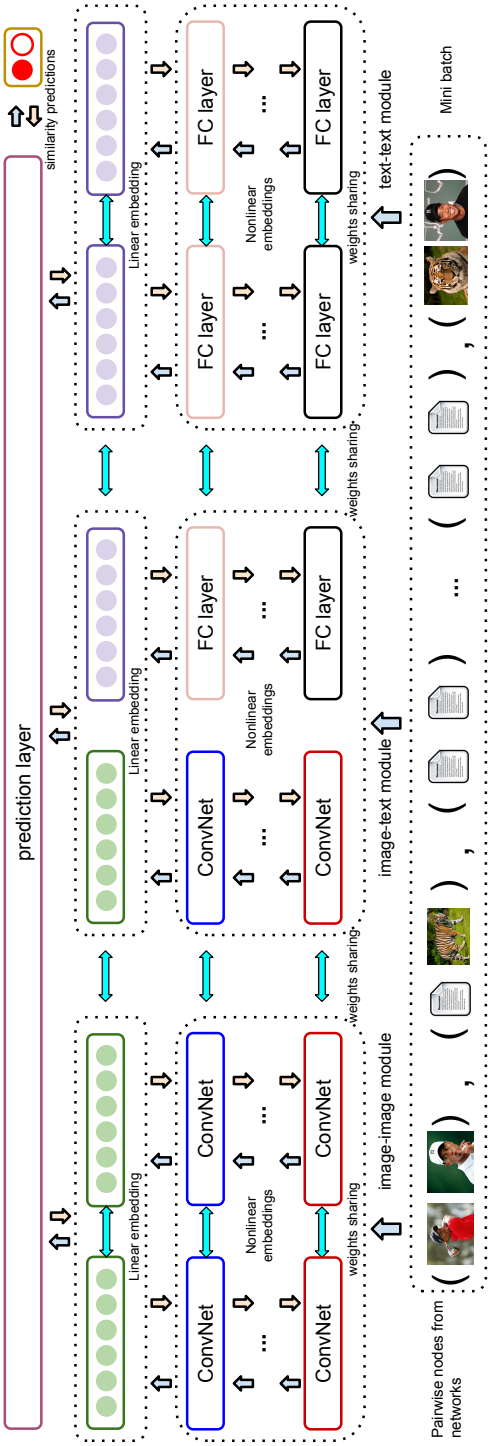


Figure 4.4: The overall architecture of HNE . The same color indicates the shared weights. The arrows are directions of forward feeding and back propagation.

4.3.1 Optimization

The objective function in equation (4.17) can be efficiently minimized by Stochastic Gradient Descent by sampling mini-batches from the training set. The advantage of training stochastically is that each mini-batch can be loaded onto GPU and computed in a parallel scheme if needed. Popular open-source deep learning packages using GPU-based implementations include Cuda-convnet [92], Caffe [118] and Theano [119], *etc.* For each input image pair, the gradient of $\mathcal{D}'_I$ is given as follows:

$$\begin{aligned}\frac{\partial(\cdot)}{\partial\mathcal{D}'_I} &= \frac{\partial(\cdot)}{\partial\tilde{p}_{\mathcal{D}'_I}(X_i)}\frac{\partial\tilde{p}_{\mathcal{D}'_I}(X_i)}{\partial\mathcal{D}'_I} + \frac{\partial(\cdot)}{\partial\tilde{p}_{\mathcal{D}'_I}(X_j)}\frac{\partial\tilde{p}_{\mathcal{D}'_I}(X_j)}{\partial\mathcal{D}'_I} \\ &= c_{ij}\cdot\left(\tilde{p}_{\mathcal{D}'_I}(X_i)\frac{\partial\tilde{p}_{\mathcal{D}'_I}(X_j)}{\partial\mathcal{D}'_I} + \tilde{p}_{\mathcal{D}'_I}(X_j)\frac{\partial\tilde{p}_{\mathcal{D}'_I}(X_i)}{\partial\mathcal{D}'_I}\right),\end{aligned} \tag{4.19}$$

where $c_{ij} = \frac{-A_{i,j}}{1+e^{A_{i,j}\tilde{p}_{\mathcal{D}'_I}(X_i)^T\tilde{p}_{\mathcal{D}'_I}(X_i)}}$. It is worth mentioning that the summation from both X_i and X_j parts is because we tie the parameters of each image module within the image-image subnetwork (symmetric to pairwise inputs). Moreover, the gradients $\frac{\partial\tilde{p}_{\mathcal{D}'_I}(X_i)}{\partial\mathcal{D}'_I}$ and $\frac{\partial\tilde{p}_{\mathcal{D}'_I}(X_j)}{\partial\mathcal{D}'_I}$ are dependent only on the structure of the deep neural network. In other words, once the deep architecture has been fixed, their gradients are automatically defined. Furthermore, for each input text pair, the gradient is similar except for changing the input and network parameters in equation (4.19) to those of the corresponding text case.

We can see that image-image inputs only contribute to learning discriminative representations for image modules. On the other hand, the cross-model inputs will affect the learning specific to both image and text. Their gradients are shown respectively as follows:

$$\frac{\partial(\cdot)}{\partial\mathcal{D}'_I} = \frac{-A_{i,j}\tilde{p}_{\mathcal{D}'_T}(z_j)}{1+e^{A_{i,j}\tilde{p}_{\mathcal{D}'_I}(X_i)^T\tilde{p}_{\mathcal{D}'_T}(z_j)}}\cdot\frac{\partial\tilde{p}_{\mathcal{D}'_I}(X_i)}{\partial\mathcal{D}'_I}, \tag{4.20}$$

and

$$\frac{\partial(\cdot)}{\partial\mathcal{D}'_T} = \frac{-A_{i,j}\tilde{p}_{\mathcal{D}'_I}(X_i)}{1+e^{A_{i,j}\tilde{p}_{\mathcal{D}'_I}(X_i)^T\tilde{p}_{\mathcal{D}'_T}(z_j)}}\cdot\frac{\partial\tilde{p}_{\mathcal{D}'_I}(z_i)}{\partial\mathcal{D}'_T}. \tag{4.21}$$

The trained deep neural network assigns different types of data to some points in a unified space so that similarities can be directly compared. So far, we have shown the proposed embedding scheme for heterogeneous networks

with only two object types: text and images. While these two types are naturally representative in many real settings, it is conceivable to expect more than two types of inputs. The proposed methods can be easily extended to handle multiple input types by considering an individual deep module for each type of data. Then, the objective function in equation (4.17) will consider all possible pairs of input types. If there are $|\mathcal{O}|$ input types, the new objective will contain $|\mathcal{O}| + \binom{|\mathcal{O}|}{2}$ object types.

Because deep learning is highly nonlinear and non-convex, globally optimal convergence is not assured. The initialization of parameters is crucial to the final performance. The literature has shown that well-designed pre-training can significantly improve final performance even when the final task is different from the pre-training task. It is worth mentioning that the proposed embedding method is unsupervised and can be used as a pre-training step for any further fine-tuning. In other words, if we want to classify network nodes, we can either obtain final features from the embedding layer and apply off-the-shelf machine learning algorithms or we can replace the prediction layer to a soft-max layer, and then fine-tune the entire deep network to a task-specific one.

4.4 Evaluation

In this section, we evaluate our proposed algorithm on several real-world datasets for both homogeneous and heterogeneous settings. The experimental results show evidence of significant improvement over many conventional baselines.

4.4.1 Datasets and Experiment Settings

We use two publicly available datasets from real-world social sites. The first one is the BlogCatalog, which is used in [120] to select features in linked social media data. The second one is a heterogeneous dataset, which is referred to as the NUS-WIDE [121]. This dataset contains both images and text. All experiment results are averaged over five different runs. The detailed descriptions and statistics for both datasets are provided below.

Table 4.1: Detailed statistics of the BlogCatalog dataset.

	Statistics
Number of nodes	5196
Number of links	171,743
Number of classes	6
Content dimensionality	8189
Balanced classes	yes

BlogCatalog [120]: It is a social blogging site where registered users are able to categorize their blogs under predefined classes. Such categorizations are used to define class labels, whereas "following" behaviors are used to construct linkages between users. The TF-IDF features are extracted from blogs as a vector representation of each individual user. Thus, blog users are represented as different nodes of the constructed networks associated with content features. It is worth mentioning that the user blogging networks are undirected, where the co-following and co-followed relationships are the same. Some detailed statistics are summarized in table 4.1.

NUS-WIDE [121]: The dataset was originally collected by the Lab for Media Search in the National University of Singapore in the year 2009. The dataset includes 269,648 unique images with associated tags from Flickr. The total number of tags is 5,018. Additionally, there are 81 groundtruth attribute labels for each image and tag pair. Since the original dataset injected many "noise" samples that did not originally belong to any of the 81 concepts, these samples were removed. Moreover, we used the most frequent 1,000 tags as text documents and extracted their TF-IDF features. We further removed those image-text pairs that did not contain any considered words. Finally, we randomly sampled 53,844 and 36,352 image-text pairs for training and testing, respectively. We constructed a heterogeneous network as the input of our proposed framework by treating images and text as separate nodes. In total, the training network contains 107,688 nodes while the testing network has 72,704. The semantic linkages between two nodes are initially constructed if they share at least one concept. We then random sample at most 30 links per node to construct the sparse matrix L . It is worth mentioning that we only evaluate our framework in an out-of-sample manner. In other words, we ensure that the training information absolutely

does not appear in any of the testing cases.

4.4.2 Network Reconstructions

Before proceeding to evaluate the proposed method in the task of classification, clustering or retrieval, we will first provide a basic and intuitive evaluation of the quality of network linkage reconstruction to validate our assumptions. Since the goal of the proposed formulation in equation (4.17) is that a good latent embedding brings objects with links closer while it pushes objects without linkage structures further, the ideal performance of the learned model can reach perfect network linkage reconstructions using equation (4.4). We first visualize the network linkage structure of the BlogCatalog dataset by randomly selecting 500 nodes and plotting their connectivities in figure 4.5. The color of each node indicates its class. As we can see, the social "following" relationships tend to connect users with similar attributes, at least from a relative point of view. On the other hand, they are noisy from an absolute point of view, in which 59.89% of links in the entire dataset connect to nodes with different classes.

We apply the proposed algorithm to learn an embedding function while monitoring the link reconstruction accuracy as shown in figure 4.6. The stochastic learning is conducted by randomly selecting 128 pairs of nodes to use as a mini-batch. The horizontal axis indicates the index of the epoch. And each epoch contains 500 mini-batches. On average, each mini-batch can be trained in less than 0.15 seconds on a single *Nvidia Tesla K40* GPU. In figure 4.6, the reconstruction performance on each mini-batch is recorded, and the line indicates the median filtered values. As more samples have been viewed by the deep HNE learner, it is able to correctly reconstruct more than 80% of the pairwise connections as compared to the initial number of 55%. Similarities propagate through sparse links across the whole network to obtain a global consistency.

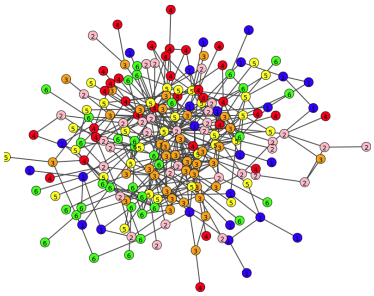


Figure 4.5: Linkage structures between 500 randomly selected nodes in the BlogCatalog dataset. The node color indicates the label of each node.

4.4.3 BlogCatalog

We first evaluate the performance of the HNE and compare it with other baselines in various tasks.

Classification:
To demonstrate the effectiveness of the representation provided by HNE, we compare our learned features with those of other feature learning methods, while keeping the classification scheme fixed. The other baseline representations are as follows:

- **Content**: Only the content feature from the original space.
- **Links**: We treat the adjacency structures as the features.
- **Link-content**: We combine features from the previous two.
- **LUFS** [120]: Unsupervised feature selection framework for linked social media data considering both content and links.
- **LCMF** [65]: A matrix co-factorization method that utilizes both linkage structure and content features.

To ensure a fair comparison, we used the same representation dimensionality and used the standard kNN classifier. In other words, the number of latent factors for LCMF is set to be the same as our output dimensionality and the first three methods are projected to a low-dimensional space using the PCA.

The average classification accuracies for the BlogCatalog dataset are shown in figure 4.7, with the output dimensionality fixed to 100. As shown, the proposed HNE method consistently outperforms other methods under different

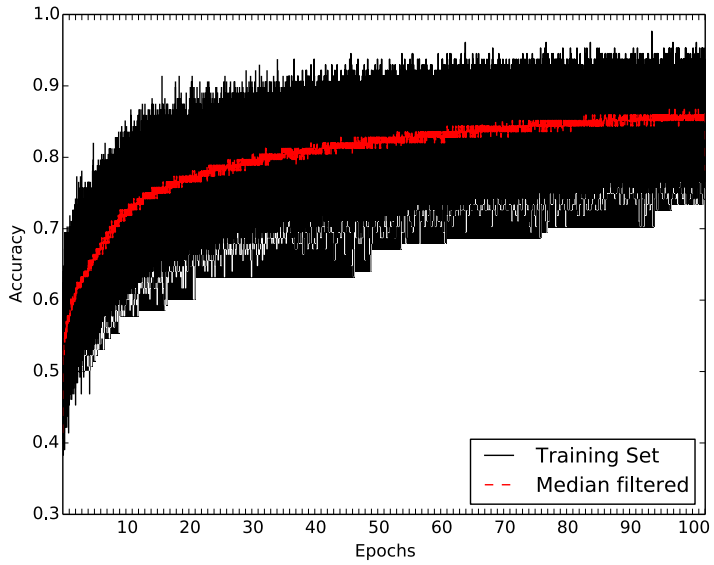


Figure 4.6: The Linkage reconstruction rate on the BlogCatalog.

training set sizes. This is because the network linkage information encodes useful insights for learning a low-dimensional embedding space by bridging linked nodes.

The rightmost bar under each setting is achieved by treating latent embedding learning as a pre-training step and fine-tuning the entire deep network by replacing the loss layer with a multi-class soft-max layer. It shows that the unsupervised latent spacing learning provides very good initializations for the supervised classification task using deep architectures and also shows that we can also achieve much higher accuracies.

Clustering:

We also compared different feature representations under the clustering task. Compared to classification, clustering is totally unsupervised, and it heavily relies on the similarity measure between different objects. We adopted the commonly used cosine similarity. The results are reported in table 4.2 using both accuracy and Normalized Mutual Information (NMI) as evaluation metrics.

The results are similar to those for the classification task. Using only links provides the worst results. This may be because, without global content information, the similarity measurements tend to be local and sensitive to

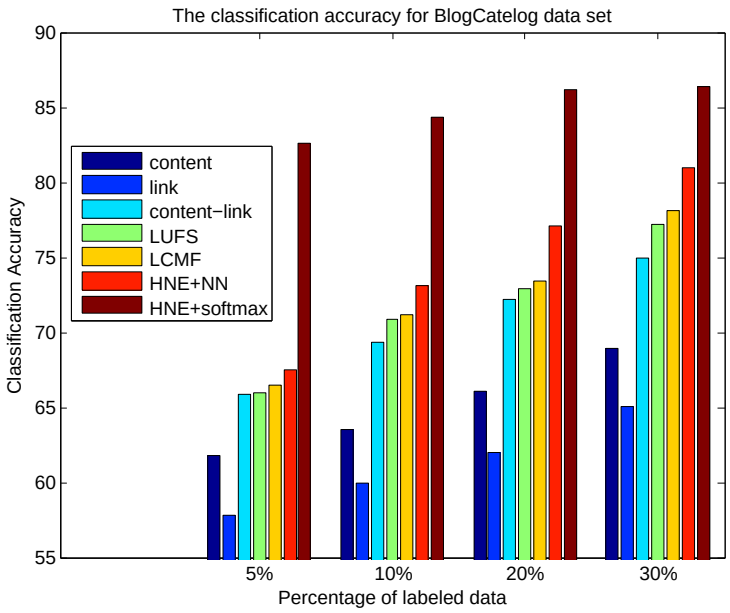


Figure 4.7: The classification accuracies among different methods under various size of training sets.

Table 4.2: The clustering result for BlogCatalog dataset.

Methods	Accuracy	NMI
content	49.06 %	0.3192
link	40.76 %	0.2482
content-link	51.69 %	0.3457
LUFS	49.88 %	0.3221
LCMF	53.91 %	0.3678
HNE	**62.37 %**	**0.4388**

noisy links. On the other hand, content similarities alone are insufficient to capture the relational knowledge. Therefore, a naive combination of the links and content provides comparable performance with other baselines. The proposed method of jointly learning the embedded space outperforms other baselines and achieves the state-of-the-art.

4.4.4 NUS-WIDE

Compared to the BlogCatalog dataset, the NUS-WIDE dataset forms a heterogeneous network that contains both images and text. We illustrate the performance of our framework for the task of classification and cross-modal

retrieval in the following subsections. Note that the latter application is not possible in the homogeneous scenario of the previous dataset.

Heterogeneous Classifications:
Given the heterogeneous scenario of this dataset, we compared our proposed method to a different set of unsupervised baselines that can specifically handle multimodal data inputs:

- **CCA**: The Canonical Correlation Analysis embeds two types of input sources into a common latent space by optimizing with respect to their correlations.
- **DTL** [122]: A transfer learning method is used to bridge semantic distances between image and text by latent embeddings.
- **LHNE**: The linear version of HNE solves the optimization function in equation (4.7).

Since our proposed method is an end-to-end learning framework, it does not require feature extraction for image inputs. We extract 4096-dimensional Cuda-convnet [92] features for all other baseline methods. The output (data in the common space) dimensionality is set to 400. Since the NUS-WIDE dataset is multi-label with unbalanced classes, we use the average precision (AP) to evaluate the classification performance for each possible label outcome. AP uses precision-recall curves for algorithmic quantification for each label. These curves are used to obtain the mean average precision (mAP). The mAP in multi-label classification domains is the standard metric which is widely used in PASCAL challenges [123] in computer vision communities. To ensure fair comparison, we use linear SVM as a common classification algorithm for all algorithms. The reason for using SVM is that calculating AP requires probabilistic interpreted confidence scores, which is inconvenient to obtain from NN classifiers.

The classification results are illustrated in table 4.3, which contains three different settings. The "image only" setting means that we learn embedding functions from the heterogeneous training set, and then train an SVM, and test classification performance on image nodes. Under the "Image + text" setting, we consider all objects in the testing network. We observe that, for all methods, categorizing text documents only is the most difficult task.

Table 4.3: The classification result in terms of mAP (mean average precision) for the NUS-WIDE dataset.

Sample	CCA	DTL	LHNE	HNE
Image only	51.96 %	52.07 %	53.16 %	**54.28 %**
Text only	51.37 %	51.88 %	51.34 %	**52.76 %**
Image + Text	52.54 %	53.22 %	53.32 %	**54.99 %**

Table 4.4: The cross-modal retrieval result (p@k) for the NUS-WIDE dataset.

Method	rank 1	rank 5	rank 10	rank 20
CCA	21.05 %	16.84 %	18.95 %	18.68 %
DTL	20.53 %	25.26 %	22.63 %	22.37 %
LHNE	26.32 %	21.05 %	21.02 %	22.27 %
HNE	**36.84 %**	**29.47 %**	**27.89 %**	**26.32 %**

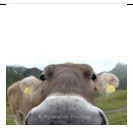
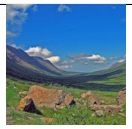
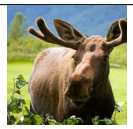
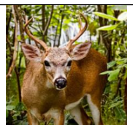
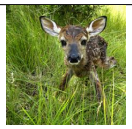
This may be because of the fact that the input text is sparse compared to images. Moreover, without the deep training, the linear version of our proposed method obtains results comparable to those of DTL which outperforms CCA. The deep architecture of HNE improves the performance further under all three settings, which demonstrates the advantage of jointly optimizing the feature learning and latent embedding with nonlinear functions.

Multimodal Search:

To further demonstrate that the learned features can be leveraged with many data mining and web search tasks, we compared our proposed method with the aforementioned baselines in the task of cross-modal retrieval. Among all 81 labels, about 75 of them appear in the TF-IDF text vector. We manually constructed 75 query vectors in the original 1000-dimensional text domain by setting the corresponding label entries to one and the remaining to zero. Using the learned embedding function, we projected these query vectors to the common latent space to retrieve all image samples in the test set using the standard Euclidean distance.

The average precision at rank k over all queries is reported in table 4.4. We observe consistent results as other tasks, and the proposed method significantly outperforms other baselines. Table 4.5 illustrates some sample retrieval results. For the query "mountain," the third retrieved result is incorrect. This might be due to the extreme visual similarities between the other mountain images and the one with a cow. The retrieval result for the

Table 4.5: Cross-model retrieval results of the proposed HNE method.

Query	**rank 1**	**rank 2**	**rank 3**	**rank 4**	**rank 5**
Mountain					
Sunset					
Cow					
Leaf					

query "cow" is not as good as the others. The first five returned images contain three deer. This is because these images have multiple labels and are connected by the concept "animal." Since our method as well as the ranking functions are totally unsupervised, these links between "deer" and "cow" objects confuse our embedding learning. We expect performance gains by using supervised ranking methods.

4.5 Conclusion

In this chapter, to alleviate the challenge brought by "variety" aspect of big data, we proposed a novel embedding scheme in networks. This approach transfers different objects in heterogeneous networks to unified vector representations. A highly nonlinear multi-layered embedding function is proposed to capture complex interactions between heterogeneous data in networks. Our approach not only simultaneously encodes network connectivity and rich content information, but also allows for similarity among cross-modal data to be measured in a common embedding space. Such a nonlinear multi-layered embedding architecture is robust, scalable and beneficial to many data mining and web search applications. Furthermore, the approach has wide applicability because a robust feature representation is useful in many big data tasks.

CHAPTER 5

HIGH-VELOCITY DATA

In this chapter, we specifically investigate the problem of similarity learning in two types of streaming networks: 1) regular networks with homogeneous node types, and 2) bipartite networks with heterogeneous node types. For generalization purposes, we do not explicitly assume any node contents and link weights that are available. We focus on these scenarios because they are closely related to the two modern applications in data mining, link prediction and recommendation with implicit feedback (also known as one-class recommendation), respectively. For example, in link prediction, networks can indicate friendships among users, while in one-class recommendation, bipartite networks can capture purchase relations between users and items.

Despite the differences in their network structures, both applications share common characteristics. First, links are in the form of positive-unlabeled (PU) measurements (*e.g.* Twitter "following", Facebook "like", Last.fm "listened" *etc.*) that do not provide negative information. Second, in the era of big data, such data are generated continuously and rapidly, and ordered temporally, determining its streaming nature. These common characteristics allow us to unify our studies into a novel framework – PU learning in streaming networks.

5.1 Streaming Network Representation

Throughout this chapter, we use the following conventions:

- Upper-cased letters, A, denote random variables/vectors. Lower-cased letters, a, denote deterministic values. Script letters, $\mathcal{A}$, denote sets.
- $p(\cdot)$ or $q(\cdot)$ denote probability density function or probability mass function, depending on whether the random variable is continuous or discrete.

- $\mathbb{E}_p(A)$ and $\mathrm{Cov}_p(A)$ denote expectation and covariance of A respectively, under the probability measure p.
- $\mathcal{N}(\mu, \Sigma)$ denotes normal distribution with mean μ and covariance Σ.
- $\{A_i\}_i$ is a set of random variables A_i with subscript i running through the index set, *i.e.* $\cup_i \{A_i\}$.

For any streaming networks, we assume there are two types of nodes: type-1 and type-2. Note that type-1 can be identical to type-2 in some problems such as link prediction. The numbers of these two types of nodes at time t are m^t and n^t respectively. The reason why they depend on t is that the proposed model accommodates the introduction of new nodes. For each type-1-type-2 node pair (i, j), L_{ij}^t is an indicator variable of whether they connect, *i.e.* it is 1 if there is an edge connecting them at time t and 0 otherwise.

This network representation is applicable to a number of real-world applications. For example, in social networks, the two types of nodes are homogeneous and represent the users, and $L_{ij}^t = 1$ represents that users i and j connect at time t, while in one-class recommendations, the two types of nodes are heterogeneous, with one representing the users and the other denoting the items, such as movies or social media posts. The two types of nodes form a bipartite graph, where $L_{ij}^t = 1$ denotes that user i interacts with (*e.g.* downloads, or gives "like" to) item j. Without the loss of generality, we assume the two types of nodes are heterogeneous. Homogeneity is merely a special case by imposing symmetry on the connections.

It is now important to emphasize the streaming nature of our setting. First, rather than presetting a fixed number of nodes, our setting allows the size of the network to be constantly and continuously changing. Second, rather than regarding the connection status between nodes as stationary, our setting regards each link status as dynamic and instantaneous. In other words, $L_{ij}^t = 1$ only reflects the connection status at that particular time t when the edge is established. It does not imply that the node pair (i, j) keeps connecting at any future times.

The instantaneity of connection status is a natural assumption for applications where the connection represents a click, a message sent *etc.*, because these interactions are themselves instantaneous. Even for applications where the connection represents some durable relation such as a "like," a Twitter "follow" *etc.*, this assumption is still reasonable because in most real-world

scenarios canceling an edge may be difficult or impossible, even though their actual connection status may have changed. For example, suppose a user gives a "like" to a Facebook post, expressing his/her interest in the post. This "like" is likely to persist even if the user's interest in the post diminishes over time. Therefore, the "like" merely reflects the user's interest at that particular moment, but hardly any time afterwards.

5.2 Streaming Positive-Unlabeled Learning - The Probabilistic Model

Our goal is to predict the node pairs that would connect in the near future, given any time t. Formally, $\forall(i,j) : L_{ij}^t = 0$, provide the prediction $\hat{L}_{ij}^{t+}$ $s.t.$ the probability of error

$$P(\hat{L}_{ij}^{t+} \neq L_{ij}^{t+}) = \mathbb{E}[(\hat{L}_{ij}^{t+} - L_{ij}^{t+})^2]$$

is minimized; where $t+$ denotes a sufficiently small amount of time after t. We adopt the standard Bayesian approach for the task, which consists of two steps: 1) model the probability distribution of L_{ij}^t; and 2) predict the connection status with a tractable inference scheme under the modeled distribution. These two steps will be detailed in this and the following section respectively. And we name the unified framework as the Streaming PU Learning, and use SPU for short.

5.2.1 Partially Observed Connection

To address the positive-unlabeled nature of our data, we assume L_{ij}^t depends on two factors: 1) The "mutual interest" of the node pair, and 2) whether their connection status is observed. In other words, the node pair (i,j) connects only when these two nodes are of interest to each other, and their connection status is observed. More concretely, in Facebook, for instance, there are two reasons that a user has not given a "thumbs up" to a post: 1) this user does not like the post at all (no mutual interests), or 2) this user has not seen the post yet (connection status unobserved).

To model the above intuition, we adapt the popular Probit model. Denote

X_{ij}^t as a real valued hidden variable modeling their mutual interest. And O_{ij}^t is the indicator variable of whether the connection status of node pair (i, j) is observed at time t. Then, the conditional distribution of L_{ij}^t is

$$L_{ij}^t = \begin{cases} 1 & \text{if } O_{ij}^t = 1 \wedge X_{ij}^t > \theta \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

The prior distributions of O_{ij}^t and X_{ij}^t are given in the following two sections respectively.

5.2.2 Connection Observability

This section proposes the prior distribution of O_{ij}^t, which denotes the connection observability. Intuitively, the connection observability is affected by several factors. The first factor is the liveliness of the network. If the amount of activity is great, *i.e.* the nodes are actively seeking links with others, then the probability of not meeting a node is low. The second factor is the cost to connect. If the cost of connection is low, *i.e.* the nodes can easily find other nodes, and can easily connect to whichever nodes they want, then the probability of having an unobserved link is also low.

While these factors are hard to evaluate explicitly, we find that the network liveliness is intuitively correlated with the number of recently established links; and the connection difficulty is inversely correlated with the number of recently born pairs, *i.e.* pairs with at least one node that is recently born. This is because the faster new content is introduced, the harder for the nodes to traverse the new content and find the nodes of interests.

Based on these intuitions, we first define the new-link-to-new-node ratio:

$$\eta^t = \frac{\#\left\{(i,j) : \exists \tau \in (t - \Delta t, t), L_{ij}^t = 1\right\}}{\#\left\{(i,j) : a_i \in (t - \Delta t, t) \vee b_j \in (t - \Delta t, t)\right\}}, \tag{5.2}$$

where Δt is a time window; a_i and b_j denote the birth times for the type-1 node i and type-2 node j respectively. Then the prior distribution of O_{ij}^t is given by

$$p\left(O_{ij}^t = 1\right) = \max\left\{\lambda \eta^t, 1\right\}, \tag{5.3}$$

where λ is a model parameter. The appropriateness of equation (5.3) will be

demonstrated in section 5.4.5.

5.2.3 Mutual Interests between Nodes

This section introduces the conditional prior of X_{ij}^t. For each type-1 node i at time t, we assume there exists a hidden topic random vector U_i^t of length r, whose elements represent type-1 node i's affinities to r hidden topics. Likewise, we use V_j^t to denote the length-r characteristic vector of type-2 node j at time t.

For each type-1-type-2 node pair (i, j), their mutual interest, X_{ij}^t, depends on the similarity between their hidden topic vectors. Intuitively, the more similar their affinity patterns are, the more mutual interests they have. Formally, the pdf of X_{ij}^t is given by

$$p\left(X_{ij}^t | U_i^t, V_j^t\right) = \mathcal{N}\left(\left(U_i^t\right)^T V_j^t, \sigma_E^2\right), \tag{5.4}$$

where σ_E^2 is a model parameter.

5.2.4 Temporal Dynamics

The hidden topic vectors of nodes tend to shift over time. To model their temporal dynamics, we assume Brownian motion:

$$\begin{aligned} p\left(U_i^t | U_i^{t-\tau}\right) &= \mathcal{N}\left(U_i^{t-\tau}, \sigma_U^2 \tau I\right) \\ p\left(V_j^t | V_j^{t-\tau}\right) &= \mathcal{N}\left(V_j^{t-\tau}, \sigma_V^2 \tau I\right). \end{aligned} \tag{5.5}$$

The initial distribution, namely the distribution at birth time, is defined differently for two distinct cases. For those nodes that are born at $t > 0$, recall that a_i and b_j denote the birth times for the type-1 node i and type-2 node j respectively. Also, for notation ease, the nodes are indexed by birth order. Then we have

$$\begin{aligned} p\left(U_i^{a_i} | \mathcal{L}^{a_i -}\right) &= \mathcal{N}\left(\frac{1}{i-1} \sum_{k=1}^{i-1} \mathbb{E}\left[U_k^{a_i} | \mathcal{L}^{a_i -}\right], \sigma_{U0}^2 I\right) \\ p\left(V_j^{b_j} | \mathcal{L}^{b_j -}\right) &= \mathcal{N}\left(\frac{1}{j-1} \sum_{k=1}^{j-1} \mathbb{E}\left[V_k^{b_j} | \mathcal{L}^{b_j -}\right], \sigma_{V0}^2 I\right), \end{aligned} \tag{5.6}$$

where $\mathcal{L}^{t-}$ is the set of observed $L_{ij}^{t'}, t' < t$, whose formal definition will be given in section 5.2.5. Equation (5.6) essentially assumes that the initial preference of a newly-born node follows the general taste of the current population, because $\mathbb{E}\left[U_k^{t-\tau}|\mathcal{L}^{t-}\right]$ and $\mathbb{E}\left[V_k^{t-\tau}|\mathcal{L}^{t-}\right]$ are MMSE estimates of the hidden topic vectors. Averaging across the whole population extracts the general topic vector.

For those nodes that exist at the beginning of the world, $t = 0$, we assume zero-mean Gaussian distribution:

$$p\left(U_i^0\right) = \mathcal{N}\left(0, \sigma_{U0}^2 I\right) \text{ and } p\left(V_j^0\right) = \mathcal{N}\left(0, \sigma_{V0}^2 I\right), \tag{5.7}$$

where σ_U^2, σ_V^2, σ_{U0}^2 and σ_{V0}^2 are model parameters.

5.2.5 The Observation Set and Events

A link L_{ij}^t is defined as an observation if and only if the following two conditions are satisfied.

Condition 1: The value of L_{ij}^t first appears or changes at time t, which involves two scenarios: 1) all L_{ij}^ts whose corresponding type-1 node i and/or type-2 node j are born at time t; 2) all L_{ij}^ts whose values jump to 1 at time t (recall that all the 1's are instantaneous as discussed in section 5.2.1).

Condition 2: For an $L_{ij}^t = 0$ to be an observation, $O_{ij}^t = 1$.

Here we would like to reiterate that the L_{ij}^ts that meet the above conditions are considered as observations at that specific time t only. Based on these two conditions, the observation set $\mathcal{L}|\mathcal{O}$ is rigorously defined by

$$\mathcal{L}|\mathcal{O} = \left\{L_{ij}^t : \left((a_i = t \vee b_j = t) \wedge O_{ij}^t = 1\right) \vee L_{ij}^t = 1\right\}, \tag{5.8}$$

where $\mathcal{O}$ denotes the set of all O_{ij}^ts. As implied by equation (5.8), the observation set is conditional on $\mathcal{O}$ which is hidden, and thus is impossible to evaluate. Following the common paradigm to marginalize over unobserved randomness, we define the unconditional observation set $\mathcal{L}$ as the set of L_{ij}^ts

that satisfy condition 1:

$$\mathcal{L} = \bigcup_{\mathcal{O}\in\{0,1\}^{|\mathcal{O}|}} \mathcal{L}|\mathcal{O} = \left\{L_{ij}^t : a_i = t \vee b_j = t \vee L_{ij}^t = 1\right\}. \tag{5.9}$$

The definition of observation set introduces our important concept of events. An event is a time instance t at which new observations are introduced. As already discussed, this includes: 1) the world starts $t = 0$; 2) a new node is introduced ($t = a_i$ or b_j); and 3) a new edge is established.

Here we define some observation- and event-related notations. For any time t, we have the following definitions:

- $\mathcal{L}$ - the unconditional observation set as in equation (5.9);
- $\mathcal{L}^t$ - the subset of $\mathcal{L}$ with time up to and including time t;
- $\mathcal{L}^{t-}$ - the subset of $\mathcal{L}$ with time up to but not including time t;
- $\mathcal{T}$ - the set of all event times;
- $\tau(t)$ - the time elapsed after the most recent (excluding current) event;
- $\tau_U(i,t)$ - the time elapsed after the most recent (excluding current) event that is related to type-1 node i;
- $\tau_V(j,t)$ - the time elapsed after the most recent (excluding current) event that is related to type-2 node j.

5.2.6 Model Summary and Joint Distribution

To sum up, the probabilistic model involves observed variables $\mathcal{L}^t$ and hidden variables $\left\{O_{ij}^t, X_{ij}^t, U_i^t, V_j^t\right\}$. The prior distributions are given by equations (5.1)-(5.7). In particular, for an event time t, the joint posterior distribution of all the hidden variables is given recursively by

$$\begin{aligned} p\left(\left\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\right\}_{i,j} | \mathcal{L}^t\right) &\propto p\left(\left\{U_i^t, V_j^t\right\}_{i,j} | \mathcal{L}^{t-\tau(t)}\right) \\ &\cdot \prod_{i,j} p\left(X_{ij}^t | U_i^t, V_j^t\right) p\left(O_{ij}^t\right) \cdot \prod_{i,j:L_{ij}^t \in \mathcal{L}|\mathcal{O}} p\left(L_{ij}^t | X_{ij}^t, O_{ij}^t\right), \end{aligned} \tag{5.10}$$

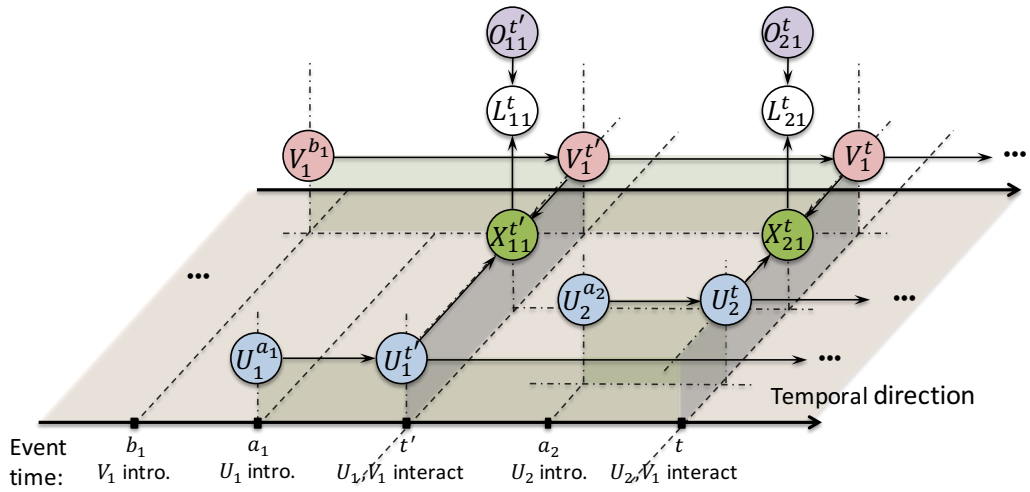


Figure 5.1: An illustrative framework of Streaming PU learning. White nodes denote observed variables, and shaded nodes denote hidden variables. The nodes are shown in a 3D coordinate grid with the time axis being the canonical x-axis as labeled. All the hidden nodes are a subset of the underlying continuous hidden process sampled at event times.

where

$$\prod_{i,j:L_{ij}^t \in \mathcal{L}|\mathcal{O}} p\left(L_{ij}^t | X_{ij}^t, O_{ij}^t\right) = \prod_{i,j:L_{ij}^t=1, L_{ij}^t \in \mathcal{L}} p\left(L_{ij}^t | X_{ij}^t, O_{ij}^t\right) \cdot \prod_{i,j:L_{ij}^t=0, L_{ij}^t \in \mathcal{L}} p\left(L_{ij}^t | X_{ij}^t, O_{ij}^t\right)^{O_{ij}^t}, \tag{5.11}$$

and

$$p(\{U_i^t, V_j^t\}_{i,j} | \mathcal{L}^{t-\tau(t)}) = p(\{U_i^{t-\tau(t)}, V_j^{t-\tau(t)}\}_{i,j} | \mathcal{L}^{t-\tau(t)}) \cdot \prod_i p(U_i^t | U_i^{t-\tau(t)}) \prod_j p(V_j^t | V_j^{t-\tau(t)}). \tag{5.12}$$

As can be seen, the last term of equation (5.11) is raised to the power of O_{ij}^t. This is merely a compact way of expressing that only those L_{ij}^ts with $O_{ij}^t = 1$ (condition 2 of being an observation) are incorporated into the joint probability distribution. An illustration of the probabilistic model is shown in figure 5.1.

5.3 Streaming Positive-Unlabeled Learning - The Model Inference

With the model established, we formulate the prediction of future connection status as a standard inference problem based on the observed connection status up to current time.

5.3.1 The Prediction Task

Our goal is to identify connections that are about to establish, *i.e.*

$$L_{ij}^t = 0, \text{ but } L_{ij}^{t+} = 1. \tag{5.13}$$

According to equation (5.1), one of the necessary conditions is to identify

$$L_{ij}^t = 0, \text{ but } X_{ij}^t > \theta. \tag{5.14}$$

The posterior expectation, *a.k.a.* the MMSE estimate, $\mathbb{E}\left[X_{ij}^t|\mathcal{L}^t\right]$, is applied to infer the hidden X_{ij}^t. However, this involves evaluating the posterior distribution as in equation (5.10), which does not bear a closed-form solution due to the complex nonlinearity of the model. Therefore, we would apply a variant of the variational approach to approximate the posterior distribution, as will be introduced in the remainder of the section.

5.3.2 Recursive Variational Inference

Approximating equation (5.10) involves two steps alternatively and recursively. First, it is approximated by the following distribution:

$$\begin{aligned} & p\left(\left\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\right\}_{i,j}|\mathcal{L}^t\right) \approx p'\left(\left\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\right\}_{i,j}|\mathcal{L}^t\right) \\ & \triangleq p'\left(\left\{U_i^t, V_j^t\right\}_{i,j}|\mathcal{L}^{t-\tau(t)}\right) p\left(X_{ij}^t|U_i^t, V_j^t\right) p\left(O_{ij}^t\right) \\ & \cdot \prod_{i,j:L_{ij}^t \in \mathcal{L}|\mathcal{O}} p\left(L_{ij}^t|X_{ij}^t, O_{ij}^t\right). \end{aligned} \tag{5.15}$$

The only difference between equations (5.10) and (5.15) is that the first term is replaced with an approximate distribution $p'(\left\{U_i^t, V_j^t\right\}_{i,j}|\mathcal{L}^{t-\tau(t)})$, which

will be defined soon.

Then, we apply the variational approximation approach to approximate $p'(\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\}_{i,j} | \mathcal{L}^t)$ as in equation (5.15) to the distribution q with the following form (for notation ease, condition on observation is omitted without causing ambiguity):

$$\begin{aligned} &p'\left(\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\}_{i,j} | \mathcal{L}^t\right) \approx q\left(\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\}_{i,j}\right) = \\ &\prod_i q(U_i^t) \prod_j q(V_j^t) \prod_{i,j} q(O_{ij}^t) q_0(X_{ij}^t | U_i^t, V_j^t)^{(1-O_{ij}^t)} q_1(X_{ij}^t)^{O_{ij}^t}. \end{aligned} \tag{5.16}$$

The key idea behind this approximation form is that when L_{ij}^t is observed, *i.e.* $O_{ij}^t = 1$, we apply the simple mean-field approximation, where each hidden variable is independent; otherwise we add the dependency on U_i^t and V_j^t to X_{ij}^t. The advantages of choosing this approximation form are twofold. First, this approximation yields a *smaller* error than the simple mean-field approximation, because the latter is merely a special case of equation (5.16) by constraining $q_0(X_{ij}^t | U_i^t, V_i^t) = q_1(X_{ij}^t)$. Second, though complicated with more dependencies, this approximation still has a tractable and concise closed-form solution.

We find the closest approximation by minimizing the KL divergence between p' and q:

$$\begin{aligned} \min_q D = \min_q \mathrm{KL}\Bigg[&q\left(\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\}_{i,j}\right) \\ &\Big\| p'\left(\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\}_{i,j} | \mathcal{L}^t\right) \Bigg]. \end{aligned} \tag{5.17}$$

Now we are ready to define p' as in equation (5.15), which depends on the q distribution at the preceding event in a similar way to equation (5.12):

$$\begin{aligned} &p'\left(\{U_i^t, V_j^t\}_{i,j} | \mathcal{L}^{t-\tau(t)}\right) \\ =&\prod_i p'\left(U_i^t | \mathcal{L}^{t-\tau(t)}\right) \prod_j p'\left(V_j^t | \mathcal{L}^{t-\tau(t)}\right), \end{aligned} \tag{5.18}$$

where

$$
\begin{aligned}
p'\left(U_i^t|\mathcal{L}^{t-\tau(t)}\right) &= \int dU_i^{t-\tau(t)} q\left(U_i^{t-\tau(t)}\right) p\left(U_i^t|U_i^{t-\tau(t)}\right) \\
p'\left(V_j^t|\mathcal{L}^{t-\tau(t)}\right) &= \int dV_j^{t-\tau(t)} q\left(V_j^{t-\tau(t)}\right) p\left(V_j^t|V_j^{t-\tau(t)}\right).
\end{aligned}
\tag{5.19}
$$

To sum up, our inference scheme can be described as:

$$
\cdots \Rightarrow q \text{ at } t-\tau(t) \Rightarrow p' \text{ at } t \Rightarrow q \text{ at } t \Rightarrow \cdots,
$$

at each **event time** t, first obtain the current p' from the q at the previous event according to equations (5.15), (5.18) and (5.19). Then, approximate the current p' with the current q according to equations (5.16) and (5.17), and so on. The q distributions are the distributions over which we perform the inference. Hence we name our inference scheme the *recursive variational inference*.

5.3.3 The Streaming Inference Scheme

In this section, we briefly state the final solution to equation (5.17). The inference scheme consists of two parts:

• Update the posterior moments of hidden variables (under $q(\cdot|\mathcal{L}^t)$) at each event time t;

• Predict future connection based on the most recent updated posterior moments.

Updating Posterior Moments:

Recall that the q distribution is the approximate distribution for the posterior distribution $p\left(\cdot|\mathcal{L}^t\right)$. Since at each event time t, the observation set $\mathcal{L}^t$ is augmented, the q distribution should be updated accordingly. Furthermore, the q distribution is characterized by its moments, and so it suffices just to update the moments. The update process is iterative: posterior moments obtained in the previous iteration are applied to update the posterior moments in the current iteration until convergence. The update equations in each iteration are given as follows.

• **The update equations for U_i^t and V_j^t:**

For a given event time t, for any type-1 node i that is involved in the event, the update equation is given by

$$\mathrm{Cov}_q\left(U_i^t\right) = \left(\Sigma_{Ui}^{-1} + \sigma_E^{-2} \sum_{j:L_{ij}^t \in \mathcal{L}^t} q\left(O_{ij}^t = 1\right) \mathbb{E}_q\left(\left(V_j^t\right)^T V_j^t\right)\right)^{-1},$$

$$\mathbb{E}_q\left(U_i^t\right) = \mathrm{Cov}_q\left(U_i^t\right)\left(\Sigma_{Ui}^{-1}\mu_{Ui} + \sigma_E^{-2} \sum_{j:L_{ij}^t \in \mathcal{L}^t} q\left(O_{ij}^t = 1\right) \mathbb{E}_q\left(V_j^t\right) \mathbb{E}_{q_1}\left(X_{ij}^t\right)\right), \tag{5.20}$$

where, for type-1 nodes that were born *before* t,

$$\Sigma_{Ui} = \mathrm{Cov}_q(U_i^{t-\tau_U(i,t)}) + \sigma_U^2 \tau_U(i,t) I, \quad \mu_{Ui} = \mathbb{E}_q(U_i^{t-\tau_U(i,t)});$$

and for type-1 nodes that were born *at* t, μ_{Ui} and Σ_{Ui} are the corresponding mean and covariance in either equation (5.7) or (5.6) depending on whether t is zero. More importantly, for nodes that are not involved in the event, no update is needed. The update equation for V_j^t is symmetric to equation (5.20) except that U and V, and subscripts i and j, are interchanged.

• **The update equations for X_{ij}^t:** For a given event time t, for any X_{ij}^t whose corresponding L_{ij}^t is in the observation set $\mathcal{L}^t$, the posterior expectation is given by

$$\mathbb{E}_{q_1}\left(X_{ij}^t\right) = \mu_{ij}^t + \left(\frac{\phi\left(e_{ij}^t\right) - \phi\left(f_{ij}^t\right)}{\Phi\left(e_{ij}^t\right) - \Phi\left(f_{ij}^t\right)}\right)\sigma_E, \tag{5.21}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ are pdf and cdf of standard Gaussian distribution respectively; and $\mu_{ij}^t = \mathbb{E}_q(U_i^t)^T \mathbb{E}_q(V_j^t)$,

$$e_{ij}^t = \begin{cases} \frac{\theta - \mu_{ij}^t}{\sigma_E} & \text{if } L_{ij}^t = 1 \\ -\infty & \text{otherwise} \end{cases}, f_{ij}^t = \begin{cases} \infty & \text{if } L_{ij}^t = 1 \\ \frac{\theta - \mu_{ij}^t}{\sigma_E} & \text{otherwise.} \end{cases}$$

• **The update equations for O_{ij}^t:**

At an event time t, if either type-1 node i or type-2 node j is involved, the update equation for O_{ij}^t is given by

$$q(O_{ij}^t = 1) = \begin{cases} 1 & \text{if } L_{ij}^t = 1 \\ \frac{p\left(O_{ij}^t = 1\right)\exp\left(\kappa_{ij}^t\right)}{1 + p\left(O_{ij}^t = 1\right)\left[\exp\left(\kappa_{ij}^t\right) - 1\right]} & \text{otherwise,} \end{cases} \tag{5.22}$$

where

$$\begin{aligned}\kappa_{ij}^t = \ln \Phi(\theta - \mathbb{E}_q(U_i^t)^T \mathbb{E}_q(V_j^t)) - \frac{1}{2\sigma_E^2} \mathrm{tr}\Big[&\mathrm{Cov}_q(U_i^t)\mathbb{E}_q(V_j^t)\mathbb{E}_q(V_j^t)^T \\ &+ \mathbb{E}_q(U_i^t)\mathbb{E}_q(U_i^t)^T \mathrm{Cov}_q(V_j^t) + \mathrm{Cov}_q(U_i^t)\mathrm{Cov}_q(V_j^t)\Big].\end{aligned} \tag{5.23}$$

In practice, we find that the trace term in equation (5.23) is often dominated, and thus is omitted to reduce computational complexity.

Predicting Future Connections:

As discussed before, for any time t and $L_{ij}^t = 0$, the prediction is based on the approximated posterior expectation of X_{ij}^t. Formally, based on equation (5.14), we would like to find those

$$L_{ij}^t = 0, \text{but } \mathbb{E}_q\left(X_{ij}^t\right) > \theta,$$

according to

$$\begin{aligned}\mathbb{E}_q\left(X_{ij}^t\right) &\approx \mathbb{E}_q\left(X_{ij}^t | O_{ij}^t = 0\right) = \mathbb{E}_q\left(U_i^t\right)^T \mathbb{E}_q\left(V_j^t\right) \\ &= \mathbb{E}_q\left(U_i^{t-\tau_U(i,t)}\right)^T \mathbb{E}_q\left(V_j^{t-\tau_V(j,t)}\right).\end{aligned} \tag{5.24}$$

Here are some intuitions. The first equality is because when there is no observation $O_{ij}^t = 0$, the posterior expectation of X_{ij}^t is equal to its prior expectation. The last equality is because U_i^t and V_j^t follow Brownian motion, and their expectations remain the same when there are no observations.

5.3.4 The Algorithm Table and Complexity

The posterior moment updating scheme is summarized in algorithm 3. In terms of computational complexity, each $L_{ij}^t \in \mathcal{L}$ appears in equation (5.20) once for each iteration; its corresponding X_{ij}^t and O_{ij}^t appear once in equations (5.21) and (5.22) respectively. Hence the total complexity over all times is $O(|\mathcal{L}|I)$, where I is the number of iterations for each update. This is a very efficient algorithm.

Algorithm 3: Streaming Posterior Update Algorithm

Input: a set of $L_{ij}^t \in \mathcal{L}$ just arrived

1 **repeat**

2 $\forall i$ involved in the current events, update $\mathbb{E}_q(U_i^t)$ and $\mathrm{Cov}_q(U_i^t)$ according to equation (5.20);

3 $\forall j$ involved in the current events, update $\mathbb{E}_q(V_j^t)$ and $\mathrm{Cov}_q(V_j^t)$ according to equation (5.20) with U and V, and subscripts i and j interchanged;

4 $\forall i, j$ pair involved in the current events, update $\mathbb{E}_{q_1}(X_{ij}^t)$ according to equation (5.21);

5 $\forall i, j$ pair involved in the current events, update $q(O_{ij}^t = 1)$ according to equation (5.22).

6 **until** *converge or maximum iteration exceed*;

7 **return** *Updated posterior moments (under q) of the hidden variables*

5.4 Evaluation

In this section, we demonstrate the practical usage of the proposed SPU framework by considering two important data mining applications: link predictions and recommendations. Our empirical studies on five real-world datasets provide strong evidence that SPU significantly improves over many state-of-the-art baselines.

5.4.1 Datasets

We utilize two link prediction and three recommendation datasets. It is worth mentioning that all five datasets are publicly available and the download links are provided. The detailed descriptions of each are listed below:

Link Predictions:

DBLP[1] [124]: This dataset is an undirected collaboration network of authors of scientific papers from the DBLP computer science bibliography. An edge between two authors represents a common publication. Edges are annotated with the date of the publication. We randomly sample 49,945 nodes among top active authors.

Epinion[2] [125]: Epinion is a popular product review site, where people can

[1] `http://konect.uni-koblenz.de/networks/dblp_coauthor`

[2] `http://www.jiliang.xyz/trust.html`

rate various products and add others members to their own trust networks or "circles of trust." Such trustworthy relationships among users are directional and represent who they may seek advice from to make decisions. We collect a total of 11,752 registered users whose in-degree and out-degree are at least one.

Recommendations:

Facebook-like Forum[3] [126]: The Facebook-like forum dataset consists of the Internet "post" activities among 899 users and 522 topics from an online community. The goal is to recommend interesting topics to candidate users that they will comment on in the near future.

MovieTweeting[4] [127]: This dataset contains the tweeting activities that consist of ratings on movies to IMDB from Twitter. Instead of predicting the specific movie ratings, we are focusing on tweeting activity itself by predicting what movie a user will rate.

Last.fm Music[5] [27]: The dataset contains the full listening history for registered users at Last.fm.[6] We only use their user ID, track ID and timestamp for the purpose of recommendations.

The statistics of the aforementioned datasets are summarized in table 5.1.

[3] http://toreopsahl.com/datasets/#online_forum_network
[4] https://github.com/sidooms/MovieTweetings
[5] http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html
[6] http://www.last.fm/

Table 5.1: Detailed statistics of the datasets. Type-1 and type-2 nodes represent users and items respectively under the recommendation setting.

	Link predictions		Recommendations		
Dataset	*DBLP*	*Epinion*	*Facebook-like*	*MovieTweeting*	*Last.fm*
Number of type-1 node	49,945	11,752	899	39,395	992
Number of type-2 node	uniform node type	uniform node type	522	22,637	107,397
Network property	undirected	directed	bipartite, heterogeneous node type		
Temporal range	12/1959-12/2013	01/2001-04/2011	05/2004-10/2004	02/2013-11/2015	02/2005-06/2009
Temporal resolution	month	second	second	second	second
Number of links	1,277,690	187,563	7,089	432,443	820,050
Sparsity	5.12×10^{-4}	1.36×10^{-3}	1.51×10^{-2}	4.85×10^{-4}	7.70×10^{-3}

Table 5.2: Performance comparison. The best performance is highlighted in bold.

	Link predictions				Recommendations					
Dataset	*DBLP*		*Epinion*		*Facebook-like*		*Twitter*		*Last.fm*	
Metric	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC
JC / ItemKNN	0.6071	0.6162	0.8066	0.8062	0.6590	0.6240	0.6381	0.6756	0.5448	0.5671
PageRank	0.6328	0.5935	0.7806	0.7118	0.7116	0.6579	0.7960	0.7282	0.7707	0.7028
OCCF	0.7093	0.6584	0.9086	0.8736	0.6959	0.6470	0.8774	0.8360	0.8616	0.8038
Time-SVD++	0.7133	0.6635	0.9398	0.8826	0.7031	0.6454	0.9167	0.8515	0.8718	0.7985
NTF	0.6920	0.6475	0.9087	0.8476	0.6378	0.6065	0.9040	0.8381	0.8597	0.7927
PUMC	0.7259	0.6660	0.9234	0.8705	0.6835	0.6404	0.9174	0.8709	0.8686	**0.8094**
SPU	**0.7412**	**0.6756**	**0.9534**	**0.8995**	**0.7339**	**0.6812**	**0.9495**	**0.8878**	**0.8831**	0.8066

5.4.2 Baseline Methods

We compare our proposed framework SPU with several representative baseline algorithms as follows:

- **JC / ItemKNN**: Jaccard's coefficient (JC) and item-based k-nearest neighbor (ItemKNN) are two of the most fundamental baselines for link prediction and recommendation respectively. We include either of them based on the task that we evaluate on.

- **PageRank**: PageRank is an iterative fixed-point algorithm over graphs, which can be applied to compute "importance" scores for each node. The prediction weight for each node pair is calculated as the product of their PageRank scores [128].

- **OCCF** [129]: One-class collaborative filtering assigns weights to unlabeled data to distinguish negative examples and unlabeled positive ones.

- **Time-SVD++ with weighted sampling** [130]: It is a variant of the Time-SVD++ algorithm, since the original one specifically takes inputs as explicitly scaled form. We utilize the same "user-oriented sampling scheme" that has been adopted by OCCF [129] to alleviate the problem under PU settings.

- **NTF** [102]: Nonnegative tensor factorization handles both temporal dynamics as well as the PU inputs. However, it differs from OCCF and Time-SVD++ with weighted sampling in that it considers all missing entries as negative.

- **PUMC** [105]: PU learning for matrix completion is a state-of-the-art one-bit factorization algorithm that aims to recover possible true negative samples by using different costs in the objective for observed and unobserved entries.

In summary, JC, ItemKNN and PageRank are three conventional methods that compute affinity scores among node pairs only based on the graph topologies, while all the other four baselines leverage the PU inputs in various ways. Among these four, Time-SVD++ and NTF also incorporate the temporal factor by confidence decay and temporal aggregation, respectively. We

make use of the open-source C++ framework from GraphChi [131] for the implementation of OCCF and Time-SVD++. The graph based algorithms including JC and PageRank are publicly available from the package in [128]. Moreover, the implementations of NTF and PUMC are acquired from the original authors.

5.4.3 Experimental Settings

For the purpose of quantitative evaluations, we follow the standard online testing protocol. Given a set of time-ordered data, we divide them into two subsets along the temporal direction. We call the first one the "validation set," and the second one the "updating and testing set." An illustrative example is shown in figure 5.2, where t_0, t_V and t_E represent the starting time, the end time of validation set and the end time of the dataset, respectively. The size of validation set is chosen to be 30% of the entire dataset. In other words, t_V is the time when 30% of connections are presented.

The testing task is to predict the possible connections of the network in the next time based on the "historical" data. Specifically, we first align the reference time t_r, also considered as the "current" time, to t_V. The prediction is evaluated at time $t_r + \Delta t$, where Δt equals the smallest temporal granularity of the dataset. The only information that can be used for model update (refining/learning latent representations) is the list of connections appearing in the time interval $[t_V, t_r]$. After performance evaluation at this specific time, we then shift t_r by Δt. In other worlds, the "current" time is now $t_V + \Delta t$. Therefore, all the data generated from temporal horizon $[t_V, t_V + \Delta t]$ can be used for prediction at $t_V + 2\Delta t$. The same procedure is performed until t_r reaches end time of the dataset t_E.

It is worth mentioning that our proposed algorithm is a fully online model, which means there is no need to retrain all latent representations from the sketch when the reference time t_r shifts by Δt. For the batch baselines, we retrain the entire model every time when t_r moves, which is much less efficient. Moreover, many baselines are insufficient to consider the case of "multiple connections" or handle the temporal resolution in a very fine grid. Although our SPU algorithm explicitly considers both aforementioned issues, for fair comparison all multiple edges are merged to the one that first appears;

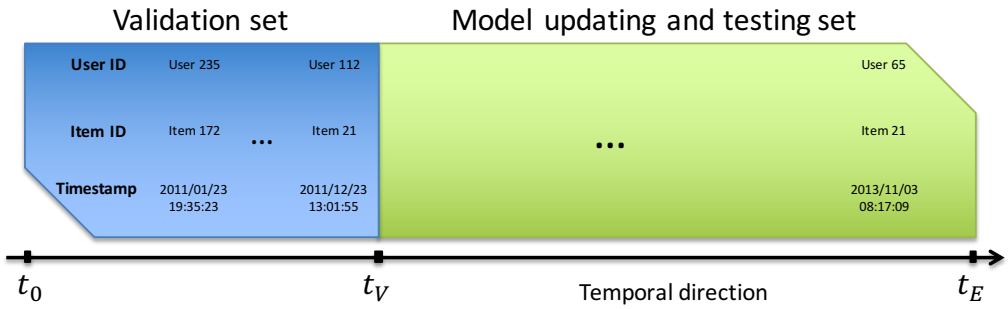


Figure 5.2: An illustrative example on data splitting. The blue region indicates the validation set while data from the green region are used for testing and model updating. At each time, the data are in a triplet format as (user ID, item ID, time-stamp).

and temporal granularity (Δt) is set to be a year for the DBLP and a week for the other four datasets.

The validation set is used to seek the best hyperparameters for each algorithm. For instance, the latent dimensionality is a model-sensitive parameter that needs to be chosen independently. Therefore, all models follow the same protocol to obtain the best set of parameters on the validation set. Once these hyperparameters are chosen, they remain the same in the testing phase.

Two commonly used metrics are suitable for both link prediction and recommendation. They are the area under the Receiver Operating Characteristic (ROC) curve (AUC) as well as the equal error rate accuracy (ACC). These two metrics are considered as classification metrics which are extremely appropriate for tasks such as "finding good objects," especially when only PU inputs are available [31, 132]. Since the evaluation task is highly imbalanced, we randomly sample the same number of negative samples (zeros) as that of positive ones at each testing time. To ensure reliability, all experimental results are averaged over 10 runs using different negative samplings.

5.4.4 Experimental Results

In this subsection, we present the empirical results of our proposed SPU framework compared to the aforementioned states-of-the-art in both link prediction and recommendation in table 5.2. We observe that the proposed algorithm consistently achieves the best performance on all five datasets. It is evident that explicitly modeling the streaming network under PU settings

significantly improves the performance for both tasks. The demonstrative features of the proposed SPU algorithm will be detailed in the following subsections.

Time-SVD++ is considered as the second best algorithm, which outperforms other baselines in three datasets. We extended the original Time-SVD++ to utilize the unlabeled data through a weighted sampling approach proposed by OCCF. The reason why Time-SVD++ outperforms not only OCCF but also other baselines is that it models the temporal information in a more suitable way. On the other hand, there is no explicit temporal consideration in OCCF. Similar to Time-SVD++, NTF also considers the temporal dynamics, but in a different way. However, its performance is even worse than OCCF for some datasets. It could be because treating all unlabeled data as negative samples hurts the performance of NTF. Another potential reason is that the algorithm considers temporal information in a retrospective way, which is inadequate to model the prospective aspect of the data streams. Moreover, PUMC obtains comparable results to Time-SVD++ across all five datasets without using any temporal information. PUMC models the PU setup in a principled way, which is able to identify the potential negative samples more accurately. At last, JC/ItemKNN and PageRank reveal the worst performance. They all belong to the standard similarity based algorithms without considering either temporal or PU characteristics of the inputs.

From the above observations, we can conclude that both temporal information and the unlabeled negativity play very important roles in the task of link prediction and recommendations. Inadequate modeling of either of these two characteristics will lead to a degradation in performance.

5.4.5 Prior Distribution Validation

In this subsection, we will examine the appropriateness of the prior of O_{ij}^t as given by equations (5.2) and (5.3). Recall that this prior is defined by our intuition that the probability of observing a connection is affected by the network liveliness and the cost of connections, which are correlated with the ratio of the number of recently established links to that of recently introduced nodes. We will apply a data-driven approach to validate this assumption.

The basic idea is that a good prior should maximize the accuracy of link prediction, and therefore we performed a greedy search to find a suboptimal path of priors across every time t that maximizes the overall prediction accuracy. Our proposed prior, as a function of event time t, will be validated if it agrees with this suboptimal path.

Specifically, the candidate values of the prior are quantized into discrete levels uniformly in the logarithmic scale from 2^0 to 2^{-17}. At each event time t, the accuracy of prediction is computed for every candidate value of the current prior, given that the priors at previous times are set to the optimal candidates in their respective greedy searches.

Figure 5.3 shows the results of this greedy search test. Each pixel of the images denotes the prediction accuracy as a function of prior candidates (horizontal axis) and event times (vertical axis). As can be seen, the yellow belt in each subplot corresponds to the prior values that yield high prediction accuracy, wherein the suboptimal path lies. The black dotted line denotes the proposed prior, which roughly follows the yellow belt of the suboptimal path. This validates our proposed prior.

5.4.6 True Negatives vs. Unlabeled

This subsection illustrates the mechanism through which O_{ij}^t deals with the positive-unlabeled data. Essentially, the key is to distinguish the true negatives from unlabeled data among all L_{ij}^ts that are 0, and place greater emphasis on the former during inference. According to the inference equation (5.20), each summation term, which corresponds to each observation at time t, is multiplied by $q(O_{ij}^t = 1)$ as weights. We will determine whether these weights are able to discriminate between true negatives and unlabeled data.

Figure 5.4 shows the weights $q(O_{ij}^t = 1)$ on 3-by-3 subsets of two datasets. At event time t, all the L_{ij}^ts in these subsets are 0, but a portion of them turn to 1 immediately afterwards, as shown by the cells marked 1 in the leftmost plots. Therefore the observed 0 in these cells are actually unlabeled data, whereas the rest of the data (marked 0) are more likely to be true negatives.

The right panel plots the evolution of the weight matrix as iteration proceeds. The gray scale in each cell denotes the weight, and the numbers are replicates of the left panel for clarity. At first, all the weights are uniform,

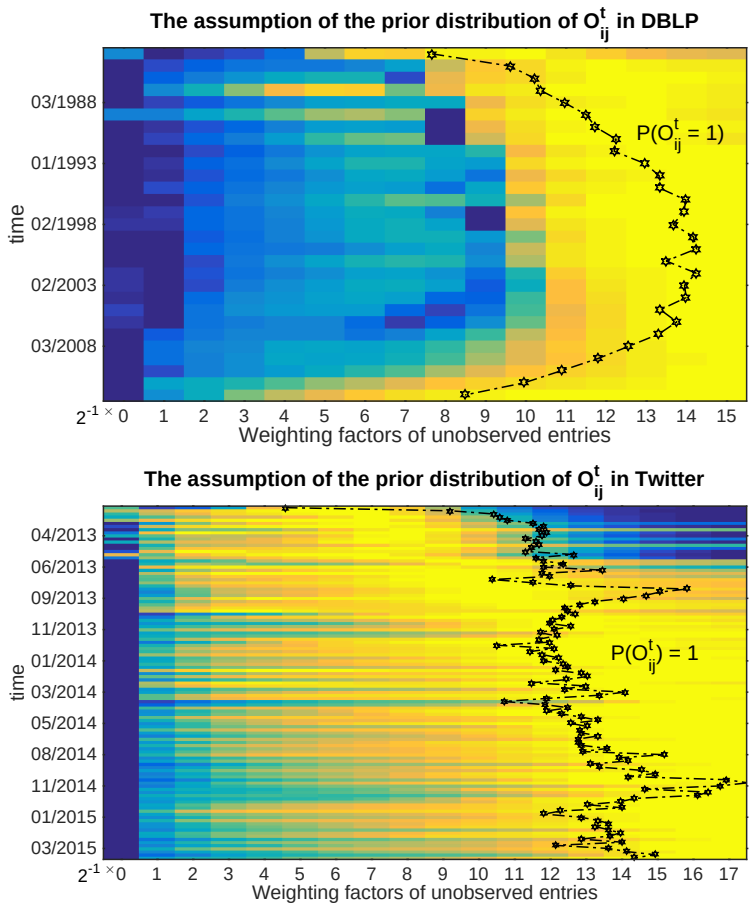


Figure 5.3: Verification of the proposed form of $q\left(O^t_{ij} = 1\right)$ (the top figure is from the DBLP, while the bottom one is the Twitter). The color in each cell denotes the accuracy if the prior is set to the candidate value. The proposed prior, denoted by dotted lines, roughly follows the high accuracy region.

where our proposed algorithm essentially reduces to many traditional link prediction algorithms that treat the all observed 0s indiscriminately as true negatives. However, upon convergence, the weights display a discriminative pattern: the weights of the unlabeled data (marked 1 as discussed) get smaller; the weights of those more probable true negatives (marked 0) become larger. In other words, during the inference iteration, the posterior distributions of the hidden topic vectors are reinforced by the data believed to be true negatives, and the interference from the unlabeled data is alleviated.

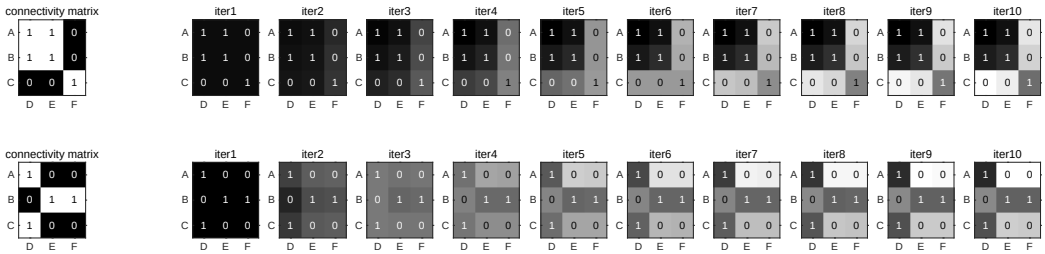


Figure 5.4: Evolution of weight matrix $q\left(O_{ij}^t = 1\right)$ (the top figure is from the DBLP, while the bottom one is the Twitter). Vertical axis denotes user, and horizontal axis denotes item. The corresponding connection statuses L_{ij}^t are all 0 at time t when the inference is performed, but a subset of them, numbered 1, soon turn to 1, and hence are originally likely to be unlabeled data. The weights, denoted by the gray scale, managed to deemphasize these unlabeled data upon convergence.

5.4.7 Temporal Drifting

Figure 5.5 and 5.6 show the evolution of averaged hidden topic vectors, *i.e.* $\mathbb{E}_q\left(U_i^t\right)$ averaged across i, and $\mathbb{E}_q\left(V_j^t\right)$ averaged across j, through time. In each plot, the left figure is for the type-1 node topic and the right for type-2. There are three observations. First, the proposed algorithm is able to capture the dynamic changes of topics, and hence can produce different predictions at different times. Second, figure 5.5 plots the result for DBLP, which is a user-user network with the two types of nodes being identical. The inference algorithm naturally yields identical topic vectors. In figure 5.6, where the two types of nodes are heterogeneous, the corresponding topic vectors are completely distinct. Third, in 5.6, we can observe a more drastic evolution in user topics (left) than in item topics (right) - there are more fluctuations in the former whereas changes in the latter are all monotonic. This agrees with our intuition that users' tastes are more volatile and influenced by trend.

5.5 Conclusion

Data in many real-world problems in the era of big data such as link prediction and one-class recommendations present similar features – positive-unlabeled and arriving at high-velocity. The common features enable us to unify a number of such problems into the novel framework – PU learning

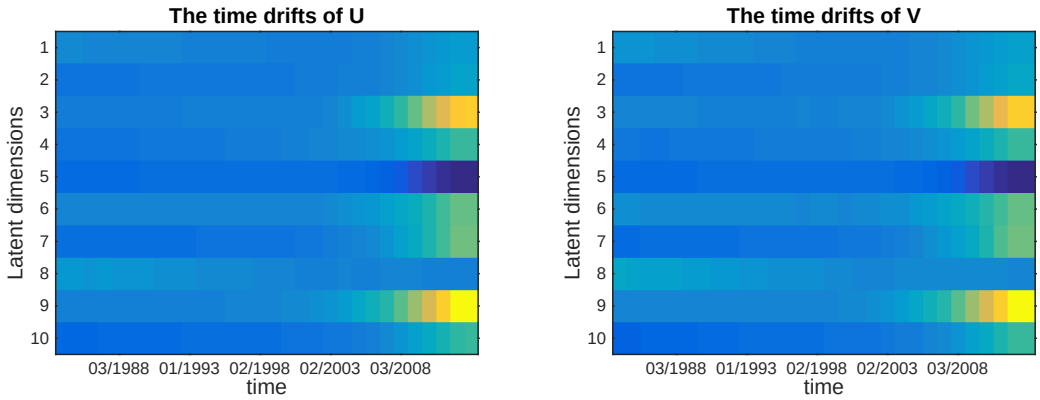


Figure 5.5: The evolution of the averaged latent topics over time in DBLP (type-1 = type-2 = user) dataset.

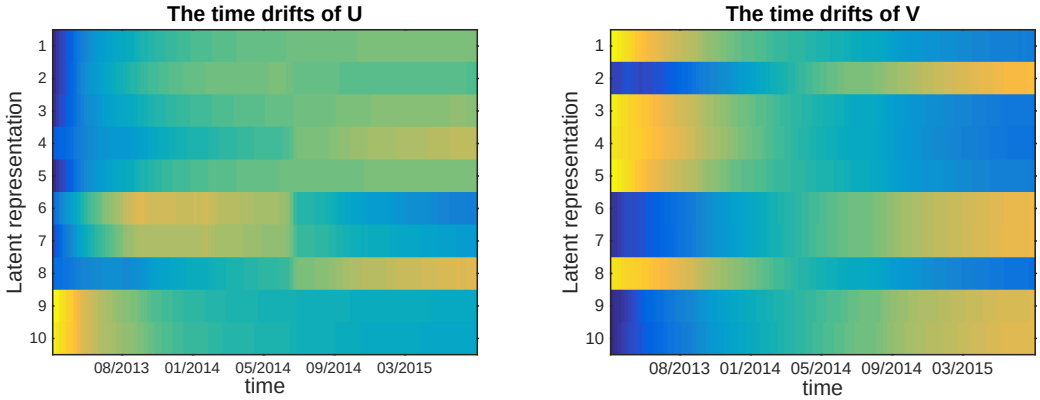


Figure 5.6: The evolution of the averaged latent topics over time in Twitter (type-1: user, type-2: item) dataset.

in streaming networks. We delineate three challenges in the problem, *i.e.*, streaming nature, unlabeled negativity and concept shift, and then propose a PU learning algorithm termed SPU that provides a principled and efficient solution to address these challenges simultaneously. We conducted experiments on various real-world datasets and experimental results suggesting that SPU can significantly advance the tasks of link prediction and recommendations.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this chapter, we summarize our research results and their broader impacts followed by a discussion on promising future research directions.

6.1 Summary of Contributions

Similarity learning as a long-lasting research problem faces many new challenges as we enter into the age of big data. In this dissertation, we investigate three major research challenges in similarity learning posed by the coherent characteristics of big data: (1) large volume; (2) data variety and (3) high velocity. For the purpose of the Ph.D. studies, we restrict our regime in social media data, which is one of the most typical big data media. Due to the natural existences of social connections in social media, we explicitly consider all data to be formed as networks.

For learning similarity in large networks, we first investigate the notion of being similar characterized by different components of networks, *i.e.*, network connectivity and node content. We reveal that with the aid of a small amount of task-specific supervision, similarities can be measured more accurately. In addition, the proposed method is able to (1) ameliorate the impact of the noisy nature of social media data and (2) accommodate massive data by distributed learning and efficient optimizations.

For learning similarity in heterogeneous networks, we propose an embedding scheme that transfers different objects to unified vector representations. The successful experiences of applying the proposed method in various data mining applications suggests that (1) maximizing homophily similarity improves the discriminative ability of the learned representations; (2) the deep embedding framework is better at capturing complex interactions between heterogeneous data; (3) the performance of the task-independent network em-

bedding framework can be further improved by viewing it as a pre-training process.

For learning similarity in streaming networks, a principled probabilistic framework is proposed to unify many big data applications to a single streaming PU learning task. We first propose a novel streaming network representation to model the data evolving process. For effective prediction, we jointly consider two major challenges under streaming settings, unlabeled negativity and concept shift, which leads to significant performance improvement in link prediction and recommendation.

In all, this dissertation investigates emerging problems and reveals novel solutions. The problem of similarity learning for large-scale, noisy, heterogeneous and high-velocity networks is challenging due to difference in data generation assumptions compared to conventional settings. Furthermore, methodologies and techniques presented in this dissertation have broad impacts. Similarity learning in networks is one the most fundamental and essential problems not only in social media data, but also in the fields of psychology, social sciences, biology, *etc.* The proposed solutions are general and applicable to many fields and data types.

6.2 Future Research

Learning similarity in big data is still in its early stages of development and an active area of exploration. Our current research raises a number of potentially challenging and promising directions that we would like to address. Examples of these areas include:

Information Trust, Transfer and Fusion: The majority of current research aims to tackle the big data problems from one or a few kinds of information sources. However, the real-world social computing paradigm is constructed by the crowd-sourced information. The overarching goal is to aggregate crowd-sourced information from multiple social and information networks to produce task-specific predictions. To achieve this goal, we aim to investigate the following related problems: (1) the source trustworthiness that aims to distinguish the untrustworthy sources from the trustworthy ones; (2) social signal processing that aims to aggregate the multi-source contributed information to recover the true signals behind the problems; (3)

the social dependency that reveals the mutual influences among different sources; and (4) the nature of information structure. With solutions of these aforementioned questions, we will be able to build a principled platform for big data analytics.

Signed Networks: We have shown many examples of solving challenges posed by big networks. However, emerging online social networks in the era of big data contain both positive and negative links, which fundamentally differ from the conventional networks under the PU setting. Since mining signed networks is in a very early stage of development, we would like to investigate the fundamental properties from the point of view of computational social science. We aim to validate whether well-known network principles (*e.g.* the power law) are still applicable. The goal is to systematically understand the nature of signed networks and establish the fundamental principles and models and to explore numerous applications.

Big Health: Fast-growing biomedical and healthcare data have encompassed multiple scales ranging from molecules, to individuals, to populations and have connected various entities in healthcare systems with increasing bandwidth, depth, and resolution. Those data are becoming an enabling resource for accelerating basic science discoveries and facilitating evidence-based clinical supports. Meanwhile, similarity learning plays an important role in clinical decision making. For instance, doctors retrieve the most similar clinical pathway for auxiliary diagnosis. However, the sheer volume and complexity of the data present major barriers toward their translation into effective clinical actions, which is definitely worth exploring in more depth.

REFERENCES

[1] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[2] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[3] A. Bellet, A. Habrard, and M. Sebban, "A survey on metric learning for feature vectors and structured data," *arXiv preprint arXiv:1306.6709*, 2013.

[4] J. Gantz and D. Reinsel, "Extracting value from chaos," *IDC iview*, vol. 1142, pp. 1–12, 2011.

[5] M. Chen, S. Mao, and Y. Liu, "Big data: a survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.

[6] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[7] P. Zikopoulos, C. Eaton et al., *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media, 2011.

[8] S. C. H. Hoi, W. Liu, and S.-F. Chang, "Semi-supervised distance metric learning for collaborative image retrieval," in *CVPR*. IEEE Computer Society, 2008.

[9] Z. Li, S. Chang, F. Liang, T. S. Huang, L. Cao, and J. R. Smith, "Learning locally-adaptive decision functions for person verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3610–3617.

[10] C. C. Aggarwal, "Towards systematic design of distance functions for data mining applications," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 9–18.

[11] S. Chang, G.-J. Qi, J. Tang, Q. Tian, Y. Rui, and T. S. Huang, "Multimedia lego: Learning structured model by probabilistic logic ontology tree," in *2013 IEEE 13th International Conference on Data Mining*. IEEE, 2013, pp. 979–984.

[12] S. Chang, C. C. Aggarwal, and T. S. Huang, "Learning local semantic distances with limited supervision," in *2014 IEEE International Conference on Data Mining*. IEEE, 2014, pp. 70–79.

[13] S. Chang, G.-J. Qi, C. C. Aggarwal, J. Zhou, M. Wang, and T. S. Huang, "Factorized similarity learning in networks," in *2014 IEEE International Conference on Data Mining*. IEEE, 2014, pp. 60–69.

[14] S. Chang, J. Zhou, P. Chubak, J. Hu, and T. S. Huang, "A space alignment method for cold-start tv show recommendations," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, 2015, pp. 3373–3379.

[15] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 119–128.

[16] J. Goldberger, G. E. Hinton, S. T. Roweis, and R. Salakhutdinov, "Neighbourhood components analysis," in *Advances in neural information processing systems*, 2004, pp. 513–520.

[17] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, no. Feb, pp. 207–244, 2009.

[18] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," in *Advances in neural information processing systems*, 2003, pp. 505–512.

[19] Z. Ou and Y. Zhang, "Probabilistic acoustic tube: a probabilistic generative model of speech for speech analysis/synthesis," in *AISTATS*, 2012, pp. 841–849.

[20] Y. Zhang, Z. Ou, and M. Hasegawa-Johnson, "Improvement of probabilistic acoustic tube model for speech decomposition," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 7929–7933.

[21] S. Chang, Y. Zhang, J. Tang, D. Yin, Y. Chang, M. A. Hasegawa-Johnson, and T. S. Huang, "Positive-unlabeled learning in streaming networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 755–764.

[22] P. Wang, B. Xu, Y. Wu, and X. Zhou, "Link prediction in social networks: the state-of-the-art," *Science China Information Sciences*, vol. 58, no. 1, pp. 1–38, 2015.

[23] G.-J. Qi, C. C. Aggarwal, and T. Huang, "Community detection with edge content in social media networks," in *2012 IEEE 28th International Conference on Data Engineering*. IEEE, 2012, pp. 534–545.

[24] A. Y. Wu, M. Garland, and J. Han, "Mining scale-free networks using geodesic clustering," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 719–724.

[25] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3, pp. 75–174, 2010.

[26] S. Chang, Y. Zhang, J. Tang, D. Yin, Y. Chang, M. A. Hasegawa-Johnson, and T. S. Huang, "Streaming recommender systems," *arXiv preprint arXiv:1607.06182*, 2016.

[27] Ò. Celma Herrada, "Music recommendation and discovery in the long tail," 2009.

[28] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[29] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *AAAI/IAAI*, 2002, pp. 187–192.

[30] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013.

[31] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou, "Recommender systems," *Physics Reports*, vol. 519, no. 1, pp. 1–49, 2012.

[32] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: bringing order to the web," Stanford InfoLab, Tech. Rep., 1999.

[33] D. Yin, Y. Hu, J. Tang, T. D. Jr., M. Zhou, H. Ouyang, J. Chen, C. Kang, H. Deng, C. Nobata, J. Langlois, and Y. Chang, "Ranking relevance in yahoo search," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 323–332.

[34] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv:1301.3781*, 2013.

[35] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, vol. 14, 2014, pp. 1532–43.

[36] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, pp. 415–444, 2001.

[37] H. Deng, J. Han, B. Zhao, Y. Yu, and C. X. Lin, "Probabilistic topic models with biased propagation on heterogeneous information networks," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 1271–1279.

[38] Y. Bengio, J.-F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet, "Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering," *Advances in neural information processing systems*, vol. 16, pp. 177–184, 2004.

[39] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Who is tweeting on twitter: human, bot, or cyborg?" in *Proceedings of the 26th annual computer security applications conference*. ACM, 2010, pp. 21–30.

[40] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, "Learning a mahalanobis metric from equivalence constraints," *Journal of Machine Learning Research*, vol. 6, no. Jun, pp. 937–965, 2005.

[41] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 209–216.

[42] G.-J. Qi, J. Tang, Z.-J. Zha, T.-S. Chua, and H.-J. Zhang, "An efficient sparse metric learning in high-dimensional space via l 1-penalized log-determinant regularization," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 841–848.

[43] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[44] W. Xi, E. A. Fox, W. Fan, B. Zhang, Z. Chen, J. Yan, and D. Zhuang, "Simfusion: measuring similarity using unified relationship matrix," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2005, pp. 130–137.

[45] Z. Lin, I. King, and M. R. Lyu, "Pagesim: A novel link-based similarity measure for the world wide web," in *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*. IEEE Computer Society, 2006, pp. 687–693.

[46] F. Geerts, H. Mannila, and E. Terzi, "Relational link-based ranking," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*. VLDB Endowment, 2004, pp. 552–563.

[47] G. Jeh and J. Widom, "Simrank: a measure of structural-context similarity," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 538–543.

[48] P. Zhao, J. Han, and Y. Sun, "P-rank: a comprehensive structural similarity measure over information networks," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 553–562.

[49] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts," in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 19–26.

[50] X. Zhu, Z. Ghahramani, J. Lafferty et al., "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*, vol. 3, 2003, pp. 912–919.

[51] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," *Advances in neural information processing systems*, vol. 16, no. 16, pp. 321–328, 2004.

[52] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *Journal of machine learning research*, vol. 7, no. Nov, pp. 2399–2434, 2006.

[53] X. Zhu, J. Kandola, Z. Ghahramani, and J. D. Lafferty, "Nonparametric transforms of graph kernels for semi-supervised learning," in *Advances in neural information processing systems*, 2004, pp. 1641–1648.

[54] X. Zhu, "Semi-supervised learning literature survey," University of Wisconsin, Tech. Rep., 2005.

[55] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2002, pp. 253–260.

[56] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme, "Learning attribute-to-feature mappings for cold-start recommendations," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, 2010, pp. 176–185.

[57] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in neural information processing systems*, 2007, pp. 1257–1264.

[58] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.

[59] P. Forbes and M. Zhu, "Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation," in *Proceedings of the fifth ACM conference on Recommender systems*. ACM, 2011, pp. 261–264.

[60] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu, "A survey of heterogeneous information network analysis," *arXiv preprint arXiv:1511.04854*, 2015.

[61] C. Wang, R. Raina, D. Fong, D. Zhou, J. Han, and G. Badros, "Learning relevance from heterogeneous social network and its application in online targeting," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 2011, pp. 655–664.

[62] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens, "Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation," *IEEE Transactions on knowledge and data engineering*, vol. 19, no. 3, pp. 355–369, 2007.

[63] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 992–1003, 2011.

[64] C. Shi, X. Kong, Y. Huang, S. Y. Philip, and B. Wu, "Hetesim: A general framework for relevance measure in heterogeneous networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 10, pp. 2479–2492, 2014.

[65] S. Zhu, K. Yu, Y. Chi, and Y. Gong, "Combining content and link for classification using matrix factorization," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 487–494.

[66] A. Paccanaro and G. E. Hinton, "Learning distributed representations of concepts using linear relational embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 2, pp. 232–244, 2001.

[67] 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*. ACM, 2009, pp. 927–936.

[68] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proceedings of the 28th international conference on machine learning*, 2011, pp. 809–816.

[69] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 650–658.

[70] Z. Yuan, J. Sang, Y. Liu, and C. Xu, "Latent feature learning in social media network," in *Proceedings of the 21st ACM international conference on Multimedia*. ACM, 2013, pp. 253–262.

[71] C. Faloutsos, T. G. Kolda, and J. Sun, "Mining large graphs and streams using matrix and tensor tools," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2007, p. 1174.

[72] J. Sun, D. Tao, and C. Faloutsos, "Beyond streams and graphs: dynamic tensor analysis," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 374–383.

[73] P. Zhao, C. C. Aggarwal, and G. He, "Link prediction in graph streams," in *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, 2016, pp. 553–564.

[74] L. S. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, and C. Sohler, "Counting triangles in data streams," in *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2006, pp. 253–262.

[75] A. L. Buchsbaum, R. Giancarlo, and J. R. Westbrook, "On finding common neighborhoods in massive graphs," *Theoretical Computer Science*, vol. 299, no. 1, pp. 707–718, 2003.

[76] A. D. Sarma, S. Gollapudi, and R. Panigrahy, "Estimating pagerank on graph streams," *Journal of the ACM (JACM)*, vol. 58, no. 3, p. 13, 2011.

[77] C. C. Aggarwal, Y. Zhao, and S. Y. Philip, "On clustering graph streams," in *Proceedings of the SIAM International Conference on Data Mining*. SIAM, 2010, pp. 478–489.

[78] C. C. Aggarwal, Y. Zhao, and S. Y. Philip, "Outlier detection in graph streams," in *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 2011, pp. 399–409.

[79] C. C. Aggarwal, Y. Li, P. S. Yu, and R. Jin, "On dense pattern mining in graph streams," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 975–984, 2010.

[80] C. C. Aggarwal, *Data streams: models and algorithms*. Springer, 2007, vol. 31.

[81] L. Golab and M. T. Özsu, "Issues in data stream management," *ACM Sigmod Record*, vol. 32, no. 2, pp. 5–14, 2003.

[82] A. McGregor, "Graph stream algorithms: a survey," *ACM SIGMOD Record*, vol. 43, no. 1, pp. 9–20, 2014.

[83] N. K. Ahmed, J. Neville, and R. Kompella, "Network sampling: From static to streaming graphs," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 8, no. 2, p. 7, 2014.

[84] Z. Wen, W. Yin, and Y. Zhang, "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm," *Mathematical Programming Computation*, vol. 4, no. 4, pp. 333–361, 2012.

[85] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[86] H. Ma, H. Yang, M. R. Lyu, and I. King, "Sorec: social recommendation using probabilistic matrix factorization," in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 931–940.

[87] S. Purushotham, Y. Liu, and C.-C. J. Kuo, "Collaborative topic regression with social matrix factorization for recommendation systems," *arXiv preprint arXiv:1206.4684*, 2012.

[88] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 448–456.

[89] T. L. Paine, P. Khorrami, W. Han, and T. S. Huang, "An analysis of unsupervised pre-training in light of recent advances," *arXiv preprint arXiv:1412.6597*, 2014.

[90] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[91] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle et al., "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[92] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[93] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[94] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 807–814.

[95] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang, "Seq-nms for video object detection," *arXiv preprint arXiv:1602.08465*, 2016.

[96] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[97] G. P. C. Fung, J. X. Yu, H. Lu, and P. S. Yu, "Text classification without negative examples revisit," *IEEE transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 6–20, 2006.

[98] H. Yu, J. Han, and K.-C. Chang, "Pebl: Web page classification without negative examples," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 1, pp. 70–81, 2004.

[99] C. Elkan and K. Noto, "Learning classifiers from only positive and unlabeled data," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 213–220.

[100] B. Liu, W. S. Lee, P. S. Yu, and X. Li, "Partially supervised classification of text documents," in *Machine Learning, Proceedings of the Nineteenth International Conference*, 2002, pp. 387–394.

[101] L. Charlin, R. Ranganath, J. McInerney, and D. M. Blei, "Dynamic poisson factorization," in *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 2015, pp. 155–162.

[102] J. Kim and H. Park, "Fast nonnegative tensor factorization with an active-set-like method," in *High-Performance Scientific Computing*. Springer, 2012, pp. 311–326.

[103] M. A. Davenport, Y. Plan, E. van den Berg, and M. Wootters, "1-bit matrix completion," *Information and Inference*, vol. 3, no. 3, pp. 189–223, 2014.

[104] T. Cai and W.-X. Zhou, "A max-norm constrained minimization approach to 1-bit matrix completion," *Journal of Machine Learning Research*, vol. 14, no. 1, pp. 3619–3647, 2013.

[105] C.-J. Hsieh, N. Natarajan, and I. S. Dhillon, "Pu learning for matrix completion," in *Proceedings of The 32nd International Conference on Machine Learning*, 2015, pp. 2445–2453.

[106] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *2008 Eighth IEEE International Conference on Data Mining*. Ieee, 2008, pp. 263–272.

[107] V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic, "One-class matrix completion with low-density factorizations," in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 1055–1060.

[108] Y. Ying, K. Huang, and C. Campbell, "Sparse metric learning via smooth optimization," in *Advances in neural information processing systems*, 2009, pp. 2214–2222.

[109] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[110] W. Cheney and A. A. Goldstein, "Proximity maps for convex sets," *Proceedings of the American Mathematical Society*, vol. 10, no. 3, pp. 448–450, 1959.

[111] S.-P. Han, "A successive projection method," *Mathematical Programming*, vol. 40, no. 1-3, pp. 1–14, 1988.

[112] C. Zeng, Y. Jiang, L. Zheng, J. Li, L. Li, H. Li, C. Shen, W. Zhou, T. Li, B. Duan, M. Lei, and P. Wang, "Fiu-miner: A fast, integrated, and user-friendly system for data mining in distributed environment," in *SIGKDD*, 2013, pp. 1506–1509.

[113] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[114] V. Vapnik, *The nature of statistical learning theory.* Springer Science & Business Media, 2013.

[115] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: Primal estimated sub-gradient solver for svm," *Mathematical programming*, vol. 127, no. 1, pp. 3–30, 2011.

[116] P. Tseng, "Convergence of a block coordinate descent method for non-differentiable minimization," *Journal of optimization theory and applications*, vol. 109, no. 3, pp. 475–494, 2001.

[117] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.

[118] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv:1408.5093*, 2014.

[119] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: A cpu and gpu math compiler in python," in *Proc. 9th Python in Science Conf*, 2010, pp. 1–7.

[120] J. Tang and H. Liu, "Unsupervised feature selection for linked social media data," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2012, pp. 904–912.

[121] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "Nus-wide: a real-world web image database from national university of singapore," in *Proceedings of the ACM international conference on image and video retrieval.* ACM, 2009, p. 48.

[122] G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Transfer learning of distance metrics by cross-domain metric sampling across heterogeneous spaces," in *Proceedings of the SIAM International Conference on Data Mining.* SIAM, 2012, pp. 528–539.

[123] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[124] J. Kunegis, "Konect: the koblenz network collection," in *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 2013, pp. 1343–1350.

[125] J. Tang, H. Gao, and H. Liu, "mtrust: discerning multi-faceted trust in a connected world," in *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 2012, pp. 93–102.

[126] T. Opsahl, "Triadic closure in two-mode networks: Redefining the global and local clustering coefficients," *Social Networks*, vol. 35, no. 2, pp. 159–167, 2013.

[127] S. Dooms, T. De Pessemier, and L. Martens, "Movietweetings: a movie rating dataset collected from twitter," in *Workshop on Crowdsourcing and human computation for recommender systems, CrowdRec at RecSys*, vol. 2013, 2013, p. 43.

[128] X. Wang and G. Sukthankar, "Link prediction in multi-relational collaboration networks," in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ACM, 2013, pp. 1445–1447.

[129] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 502–511.

[130] Y. Koren, "Collaborative filtering with temporal dynamics," *Communications of the ACM*, vol. 53, no. 4, pp. 89–97, 2010.

[131] A. Kyrola, G. Blelloch, and C. Guestrin, "Graphchi: large-scale graph computation on just a pc," in *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, 2012, pp. 31–46.

[132] A. K. Menon and C. Elkan, "Link prediction via matrix factorization," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2011, pp. 437–452.

APPENDIX A

PROOFS

A.1 Proof of Lemma 3.2.1

Proof $\mathcal{T}$ can be expressed as the intersection of $|\mathcal{S}|$ sets as $\mathcal{T} = T_1 \cap \cdots \cap T_{|\mathcal{S}|}$. Each T_m involves a set of triplet supervision. Without loss of generality, assume $T_m = \{S : S_{ij} \geq S_{ik} + 1\}$. It can be easily verified that T_m is a convex set by the definition of convex sets by assuming $S^1, S^2 \in T_m$, $\alpha \in [0, 1]$. Then, the following is true:

$$\begin{aligned} \alpha S^1_{ij} + (1-\alpha) S^2_{ij} &\geq \alpha(S^1_{ik} + 1) + (1-\alpha)(S^2_{ik} + 1) \\ &\geq \alpha S^1_{ik} + (1-\alpha) S^2_{ik} + 1. \end{aligned}$$

Therefore, $\alpha S^1 + (1-\alpha) S^2 \ \in \mathcal{T}_m$ and $\mathcal{T}_m$ is a convex set. Furthermore, $\mathcal{T}$ is an intersection of a finite number of convex sets. Therefore, $\mathcal{T}$ is convex. ∎

A.2 Proof of Theorem 3.2.2

Before preceding to the proof of theorem 3.2.2, we first need to prove the following lemma:

Lemma A.2.1 *For any x, y, x' and $y' \in \mathbb{R}$ such that $x' \leq y' - c$, where $c \in \mathbb{R}^+$, $x' = \frac{1}{2}(-c + x + y)$ and $y' = \frac{1}{2}(c + x + y)$ provide the minimal value of the least squares function $f(x, y, x', y') = (x - x')^2 + (y - y')^2$ if $x > y + c$. For $x \leq y - c$, the minimal $f(x, y, x', y')$ is obtained by setting $x' = x$ and $y' = y$.*

Proof The problem can be formulated as a constrained convex program as

$$\min_{x', y'} \quad (x' - x)^2 + (y' - y)^2 \quad \text{subject to: } x' \leq y' - c.$$

The optimal solution can be interpreted as numerically solving the KKT system of equations [43]. The Lagrangian dual problem is

$$\max_\lambda \ \min_{x',y'} \quad (x'-x)^2 + (y'-y)^2 + \lambda(x'-y'+c),$$

where λ is so called the KKT multiplier. The optimal x'^* and y'^* is achieved if satisfies some regularity conditions such as: the *stationarity*

$$\begin{cases} 2(x'-x)+\lambda = 0 \\ 2(y'-y)-\lambda = 0 \end{cases} \Rightarrow \begin{cases} x' = -\frac{1}{2}\lambda + x \\ y' = \frac{1}{2}\lambda + y \end{cases},$$

the *primal feasibility* $x'-y'+c \leq 0$, the *dual feasibility* $\lambda \geq 0$, and the *complementary slackness* $\lambda(x'-y'+c) = 0$. By solving the system of equations we obtain the optimal solution of x'^* and y'^* as

$$\text{if } \lambda = 0 \text{ then } \begin{cases} x'^* = x \\ y'^* = y \end{cases}, \quad \text{otherwise } \begin{cases} x'^* = (-c+x+y)/2 \\ y'^* = (c+x+y)/2. \end{cases}$$

This thus completes the proof. ∎

Now, we have all the tools to prove theorem 3.2.2 as follows:

Proof For any $S \in \mathcal{T}_m$, we have the trivial solution that the projection is itself. For any $S \notin \mathcal{T}_m$, we are seeking the optimal value of S^*, such that the projection error $\|S - S^*\|_F^2$ is minimized. In other words, the solution to the minimization problem of $\min_{S^* \in \mathcal{T}_m} \|S - S^*\|_F^2$ provides the projector. Because the Frobenius norm is decoupled for every element, it follows that $\mathcal{T}_m$ only affects the entries of S^*_{ij} and S^*_{ik}. Therefore, by choosing $S^*_{pq} = S_{pq}$, we obtain zero projection error for S^*_{pq} for all $\{p,q\} \neq \{i,j\}$ and $\{i,k\}$. The minimization problem is further reduced to the following:

$$\min_{S^*_{ij} \geq S^*_{ik}+1} (S_{ij} - S^*_{ij})^2 + (S_{ik} - S^*_{ik})^2,$$

where the property of the optimal solution is given in lemma A.2.1. This completes the proof. ∎

A.3 Proof of Theorem 3.5.1

Proof First of all, since P is a contentious function defined on a compact set specified by $0 \leq \alpha_i \leq \lambda_4$, the range of P is also a compact set, and it follows that P^* exists and $-\infty < P^* < \infty$. In the following text, we will prove that by each iteration of coordinate descent described in algorithm 2, the decline of the value of the objective function is bounded from above, from which both the convergence and the bound for the number of iterations required for convergence are established.

After the t-th ($t \geq 0$) iteration, if the algorithm goes on to the $(t+1)$-th iteration, then $\|\alpha_i^{t+1} - \alpha_i^t\|_2 \geq \varepsilon_0$. Letting $s' = \arg\max_s |\alpha_{is}^{t+1} - \alpha_{is}^t|$, it follows that $(\alpha_{is'}^{t+1} - \alpha_{is'}^t)^2 \geq \frac{\varepsilon_0^2}{|\mathcal{R}_i|}$, which is the lower bound for the maximum change of the elements of α_i in t-th iteration. Now we will consider three cases in the updating formula (3.37) to get the bound for the change of the objective function given the change of the s'-th element of α_i, i.e. $\alpha_{is'}$.

According to Taylor's theorem, we obtain

$$\begin{aligned} P(\alpha_{is'}^{t+1}) - P(\alpha_{is'}^t) = & \frac{\alpha_{is'}^t - R_{s'}(1+\lambda_1)}{1+\lambda_1}(\alpha_{is'}^{t+1} - \alpha_{is'}^t) \\ & + \frac{1}{2(1+\lambda_1)}(\alpha_{is'}^{t+1} - \alpha_{is'}^t)^2. \end{aligned} \tag{A.1}$$

When $0 \leq R_{s'}(1+\lambda_1) \leq \lambda_4$, $\alpha_{is'}^{t+1} = R_{s'}(1+\lambda_1)$, thereby the change of the objective function is

$$\begin{aligned} P(\alpha_{is'}^{t+1}) - P(\alpha_{is'}^t) &= \frac{\alpha_{is'}^t - \alpha_{is'}^{t+1}}{1+\lambda_1}(\alpha_{is'}^{t+1} - \alpha_{is'}^t) + \frac{1}{2(1+\lambda_1)}(\alpha_{is'}^{t+1} - \alpha_{is'}^t)^2 \\ &= -\frac{(\alpha_{is'}^{t+1} - \alpha_{is'}^t)^2}{1+\lambda_1} + \frac{1}{2(1+\lambda_1)}(\alpha_{is'}^{t+1} - \alpha_{is'}^t)^2 \\ &= -\frac{1}{2(1+\lambda_1)}(\alpha_{is'}^{t+1} - \alpha_{is'}^t)^2 \leq -\frac{\varepsilon_0^2}{2|\mathcal{R}_i|(1+\lambda_1)} \ . \end{aligned} \tag{A.2}$$

When $R_{s'}(1+\lambda_1) > \lambda_4$, $\alpha_{is'}^{t+1} = \lambda_4$. Also, since $0 \leq \alpha_{is'}^t \leq \lambda_4$, $\alpha_{is'}^t \leq \alpha_{is'}^{t+1}$.

The change of the objective function is

$$
\begin{aligned}
P(\alpha_{is'}^{t+1}) - P(\alpha_{is'}^{t}) &= \frac{\alpha_{is'}^{t} - \alpha_{is'}^{t+1} + \alpha_{is'}^{t+1} - R_{s'}(1+\lambda_1)}{1+\lambda_1}(\alpha_{is'}^{t+1} - \alpha_{is'}^{t}) \\
&\quad + \frac{1}{2(1+\lambda_1)}(\alpha_{is'}^{t+1} - \alpha_{is'}^{t})^2 \\
&= -\frac{1}{2(1+\lambda_1)}(\alpha_{is'}^{t+1} - \alpha_{is'}^{t})^2 + \frac{\alpha_{is'}^{t+1} - R_{s'}(1+\lambda_1)}{1+\lambda_1}(\alpha_{is'}^{t+1} - \alpha_{is'}^{t}) \\
&\leq -\frac{1}{2(1+\lambda_1)}(\alpha_{is'}^{t+1} - \alpha_{is'}^{t})^2 \leq -\frac{\varepsilon_0^2}{2|\mathcal{R}_i|(1+\lambda_1)},
\end{aligned}
\tag{A.3}
$$

since $(\alpha_{is'}^{t+1} - R_{s'}(1+\lambda_1))(\alpha_{is'}^{t+1} - \alpha_{is'}^{t}) \leq 0$.

Similarly, when $R_{s'}(1+\lambda_1) < 0$, $\alpha_{is'}^{t+1} = 0$, and $\alpha_{is'}^{t} \geq \alpha_{is'}^{t+1}$. We still have $(\alpha_{is'}^{t+1} - R_{s'}(1+\lambda_1))(\alpha_{is'}^{t+1} - \alpha_{is'}^{t}) \leq 0$ and it follows that the change of the objective function is

$$
P(\alpha_{is'}^{t+1}) - P(\alpha_{is'}^{t}) \leq -\frac{1}{2(1+\lambda_1)}(\alpha_{is'}^{t+1} - \alpha_{is'}^{t})^2 \leq -\frac{\varepsilon_0^2}{2|\mathcal{R}_i|(1+\lambda_1)}. \tag{A.4}
$$

Based on (A.2), (A.3) and (A.4), the change of the objective function given the change of $\alpha_{is'}$ is bounded from above by $-\frac{\varepsilon_0^2}{2|\mathcal{R}_i|(1+\lambda_1)}$ after the t-th iteration.

Moreover, let $P_0 = P(\alpha_i^0)$ be the initial value of the objective function; then the difference between the initial value and the optimal value of the objective function is $P_0 - P^* < \infty$. Therefore, after at most $\left\lceil \frac{P_0 - P^*}{\frac{\varepsilon_0^2}{2|\mathcal{R}_i|(1+\lambda_1)}} \right\rceil = \left\lceil \frac{2|\mathcal{R}_i|(P_0 - P^*)(1+\lambda_1)}{\varepsilon_0^2} \right\rceil$ iterations, Algorithm 2 converges. ∎